Systematic Security Comparison of Different Federated Learning Approaches

Author:
Hesham N. Y. Abdelhay
MSc Information Security

Supervisor:
Professor. Emiliano De Cristofaro
University College London

Submission Due Date: 13th November 2020

Abstract

Federated Learning is a term coined by researchers at Google for what is effectively secure and private distributed machine learning, and implemented across millions of devices. Many researchers have implemented singular attacks against this system, and developed defences for it, but the varying state of the testing architecture, and the parameters used means that these cannot always be directly comparable and the true effect of these attacks is not always apparent.

This paper presents a hybrid research and implementation paper in which the current state of Federated Learning is evaluated with respect to security and privacy. This paper then presents different secuirty attacks and custom implementations to analyse the effectiveness of these attacks on both Horizontal and Vertical Federated Learning and the associated results. This paper shows examples of comparable security attacks such as model poisoning are more effective than training data poisoning, which is also more effective than data test poisoning. Backdoor attacks are also evaluated of their effectiveness in certain condition.s

This paper contributes to the current state of research by unifying what has come before it in the context of security attacks in Federated Learning, and implementing experiments on the same system to provide comparable and quantifiable metrics of attack effectiveness and parameters to prove whether certain attacks are truly effective or not.

Acknowledgements

The Academic Year 2019/20 has been riddled with extraordinary events, and without the immediate support of the UCL Department of Computer Science I would have not been able to complete this dissertation and my master's degree in general, and therefore my greatest thanks is given to them for their efforts.

Additionally, I would like to express my gratitude for the immense support of my Father, Mother, Brother and closest friends. This has been an ongoing motivating force to complete this dissertation through the recent challenges posed to me and my academic cohort, for which I am truly grateful.

I would also like to thank my supervisor Professor Emiliano De Cristofaro for allowing me to conduct this dissertation in his respected research field, which has given me an unparalleled insight into the importance of Security and Privacy in the evolving field of Machine Learning.

Contents

1 Introduction

Artificial Intelligence (AI) is one of the fastest developing fields in technology that will genuinely change the way society functions, from decision making to human-computer interaction. AI has the capability to solve problems in our world once thought impossible. Machine Learning is considered one of the pillars of AI, in which large amounts of data is processed to find trends and patterns that could not have been found by manual analysis, where these patterns lie predictions and decisions about the most minuscule of problems.

Whilst Machine Learning seems to be adopted in all aspects of our data-driven society, the impact of this information flow must also be studied, and for humans to allow machines to drive our decision-making process, it is clear that we must increase our confidence in the models generated.

Recently, security and privacy have become a hot topic in machine learning, where the integration of security technologies and privacy policies are generally recognised as the only way to truly accept a machine learning model's validity and integrity, and as more data is now generated on internet-connected local devices, the need for scalable, distributed, secure, and private machine learning systems is clearly evident.

1.1 Motivation and Goals

The motivation of this dissertation is the real-world applicability of machine learning and respective security challenges, and therefore the aim is to evaluate the security-attack susceptibility of an extremely relevant and ever-growing machine learning framework – Federated Learning - introduced by the researchers at Google, McMahan et al. [15]. This research area is lacking "unified" papers that focus on more than one attack or experiment, and the need for a collation of ideas and testing in a "security test" of Federated Learning is the main driving force, with many disjointed papers being released that feature different attacks on different iterations of the Federated Learning system with varying parameters, hardware/software resources, and different statistical distributions of data. Therefore, the broad goals of this dissertation are to create a unified paper that contributes the following:

1. Explore available literature on Federated Learning, Security and Privacy, and collate the information into a single "unified" source of information.

2. Define a virtualised baseline Federated Learning system on which multiple attacks can be implemented and mounted, providing a comparable set of results on attack feasibility, effects and analysis for different statistical distributions of data.

3. Define "unified" attack strength parameters and nomenclature transferable across attacks and results.

4. Systematically compare the effect of security attacks and statistical distributions that are regularly seen in the wild on the attacks.

5. Define suitable future directions of research in a similar "unified" approach for other security problems, in addition to privacy attacks.

1.2 Dissertation Structure

This dissertation is organised as follows: The Second chapter (Background) paints a contextualised image of Federated Learning within the field of Machine Learning. This then follows on to an extensive literature review of different Federated Learning approaches and security/privacy attacks. Chapter 3, defines the methodology that was used to design the experiments, and key decisions on the design choices. Additionally, this section outlines all the attacks being conducted and the data being collected. Chapter 4 presents all the results of the experiments alongside analysis of the results. Chapter 5 presents a discussion about the results, in which intricacies are discussed and possible hypothesise developed. The final chapter is the conclusion which mentions limitations of the experiments and future work, in addition to final remarks.

2 Background

In order to contextualise Federated Learning in the field of Machine Learning, the following survey into the current state of research is presented:

2.1 Machine Learning

Machine Learning is a sub-set of Artificial Intelligence that operates on the premise of decision automation and using data to generate models on relationship points (or patterns) found in this data. Machines are not explicitly given logical rules on the way the data should be processed, and this task is left to the machine to effectively "learn". This basic research dates back to the mid 1900's in which the objective was to mimic a basic level of human intelligence, and to have machines intelligently learn and make decision with explicit instruction – similar to how the human brain makes them. These machine learning systems are designed using three distinct techniques defined by Papernot et al. [16] dependant on the resources and environments available:

- **Supervised Learning:** This is the most structured form of Machine Learning with the dataset containing inputs and respective labelled output. This allows the machine to determine direct relationships within the inputs and the expected output and build a model upon this. For example, a supervised machine learning model can be used to determine what the content of a picture given a dataset that includes picture and its corresponding label (e.g. a picture of a dog with the label "dog").

- **Unsupervised Learning:** This is a semi-structured form of Machine Learning with the dataset having unlabelled data inputs. The machine is then expected to determine relationships, and more importantly, trends in the data that provide meaningful models. This trend analysis is usually performed with a statistical analysis function such as clustering to group data points into classes.

- **Reinforcement Learning:** This is an extremely unstructured form of Machine Learning in which the machine must build a view of the data through what is effectively trial-and-error and self-learning. The machine is given rewards for correct actions which allow the machine to build a model of the final objective but also correct trends and relationships of the data. Arguably, this is the type of learning that most reflects the real-world and the way in which humans learn (for example, how to walk as a baby and how to write as a child) and therefore the most prone to complexities.

2.2 Models

Machine Learning models can effectively perform two distinct tasks, either learn from the data provided and output information about the learning process such as classifications and probabilities, or the model is used to generate data based on the input data. Machine learning models that classify information and effectively probabilities of trends within the data are defined as *Discriminative Models*, whereas those that generate data are adequately named *Generative Models*.

Supervised Learning is the most commonly implemented Machine Learning technique due to its simplicity and effectiveness, and discriminative models are the most commonly used model type in data science as their use-cases are arguably infinite, from solving world-wide social problems to business ideas built on ML. Within these models, different algorithms can be used to find trends with the data or build probability predictions of outcomes, and these are a set of common algorithms that are widely used in supervised learning:

- **Regression:** This is a simple trend learning model where the machine plots the data-points on a graph and builds a line-of-best-fit that can be used to predict points that are not on the graph. This can be linear with a straight line or slight more fitting with polynomial curve-of-best-fit. There is a binary classification version of regression named Logistic Regression which builds a line that essential classifies inputs to only 2 output values of either 0 or 1. These are useful for decisions such as yes/no for example.

- **Decision Trees:** These are commonly built where the data is used to make sub-decisions that ultimately lead to a final decision based on the input. Probabilities are then associated with each branch of the tree as the model learns which input leads to which output respectively. Decision trees can be small, inaccurate but fast to traverse, or large, very accurate but slow to traverse all branches.

- **Neural Networks:** This is essentially a matrix that maps inputs to output and combinations of these inputs using nodes that are linked with a concept called *weights*. Weights allow the model to associate different inputs to outputs, and combinations of these with different probabilities. There are layers in a neural network that the machine builds, and these are updated every time an input is processed to ultimately build a prediction of what the output is in comparison to the input and a corresponding accuracy percentage when the predicted output is compared to the expected output from the labelled dataset.

2.3 Secure and Private Machine Learning

As we head towards a world that commonly uses Machine Learning to perform automated tasks that could affect real people in real-world scenarios, it is imperative that the data and models are kept both secure and private. These requirements have evolved and developed Machine Learning to incorporate security and privacy preservation directly into the system architecture:

- **Secure Machine Learning:** This is a set of policies that outline what Machine Learning systems should be aware of and the security threats posed by adversaries [12]. Secure Machine Learning should be able to perform under the threat of the attacks mentioned further down this section, with varying adversarial goals. The paper by Liao, Ding, and Wang [12] includes a set of definitions in which the different types of security attacks can be grouped as:

  - **Attack Influence:**
    - *Causative Attacks:* Adversarial Attacks that attempt to influence the model with direct control over the Training Data.
    - *Exploratory Attacks:*Adversarial Attacks that attempt to influence the model without any control over the Training Data.
  - **Security Violations:**
    - *Integrity Attacks:* Adversarial Attacks that attempt to compromise the model by introducing false negatives in the training data.
    - *Availability attacks:* Adversarial Attacks that attempt to cause denial-of-service of the model by introducing false positives in the training data.
  - **Specificity of Attacks:**
    - *Targeted Attacks:* A specific model and machine is targeted by an adversary.
    - *Random Attacks:*Any model and machine is targeted randomly by an adversary.

- **Privacy Preserving Machine Learning:** Privacy is a key requirement of many users, and machine learning does not always put privacy first. Therefore, privacy preserving machine learning is a additional layer of technology to obfuscate personal information in the underlying dataset from unauthorised viewing. This can be done for example by using cyrptography to encrypt the data, or anonymisation techniques such as Differential Privacy, such that the individual data sample does not represent a persons private data, but we can still learn from the model in aggregate[5].

2.4 Federated Learning

Contextually, there is always only two ways a task can be completed - either individually, or collaboratively, and machine learning is no different. Machine learning tasks can take place on a single client, in which all the data is loaded onto the client, the model and algorithm is computed, and the results are output. On the other hand, machine learning can also be distributed, in which multiple clients train the same model on local datasets using their own computational resources with the objective of furthering the objectives of the share collaborative model. This is an extremely useful way of conducting any tasks as the famous quote goes "a problem shared is a problem halved". Whilst this may be true, it poses questions on the privacy of the data if its being seen by multiple clients and security of the system and whether the right people have access.

Federated Learning is an attempted solution at this proposed collaborative distributed machine learning, and is a term first coined by McMahan et al. [15] at Google that attempts to tackle the security and privacy issues of distributed machine learning. McMahan et al. recognises that the de-facto centralised machine learning models that use centralised datasets is risky, stating that "rich data is often privacy sensitive, large in quantity, or both".

Federated Learning was proposed to decentralise the learning process and data ownership to local clients such that the data is generated, hosted and processed on the client device, but with the intention of global collaboration.

Initially, in the paper by McMahan et al., they proposed a simple process for conducting Federating Learning in a series of update rounds (Figure 1), where not every client is required to participate as follows:

1. A subset of all clients is selected to participate in the round, and they "pull" the current state of the global model.
2. Each client that is participating in this round uses their local data to compute an update to the weightings of the local model.
3. Each participating client transmits only the recently computed model update back to the aggregation server
4. The aggregation server then collects all the model updates and averages these updates and integrates the new weightings as an updated - and by design an improved - model.

Federated Learning was introduced as a distributed client-server architecture, where a

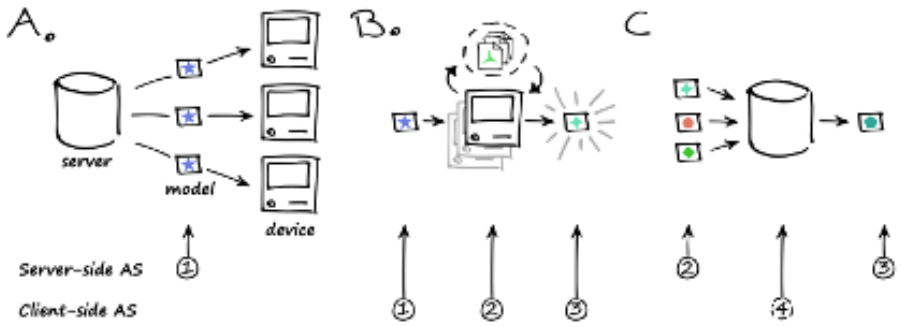

Fig. 1. *Top*: The Federated learning process: A) A model is sent from the servers to the devices. B) The devices train the model further on their local dataset. C) The devices send their model back to the server which combines them into a singular model.

Figure 1: Federated Learning Process - Enthoven and Al-Ars [6]

centralised secure server is only used for co-ordination of clients, and aggregation of processed information using a Federate Averaging algorithm. The private data is essentially set as local-access only where local machine learning updates are computed on device and updated to the global model.

It is worth noting that this updated process was then re-released by the same Federate Learning research team at Google and a follow paper was released by Konečný et al. [10] swiftly after the original paper [15]. This updated paper modifies the process of local model updates by introducing the following options:

- **Structured Updates:** Each update is essentially a defined structure, and a restricted space of a small number of parameters are included in each update to reduce communication bandwidth and data sent at each update. Konecny et al. propose two distinct structures for these updates:
  - *Low Rank Structure:* All local model updates are in the form of a Low Rank Matrix with a fixed size.
  - *Random Mask Structure:* A more complex idea that defines all local model updates to be in the form of a Sparse Matrix with a pre-defined sparsity pattern
- **Sketched Updates:** This is essentially a lossy compression algorithm that is used on each local model update. The compression algorithm encodes the data and then this is decoded on the server before the Federated Averaging algorithm is run. These lossy compression can be in the form of sub-sampling such that the average sample value is representative of the true value, or probabilistic quantisation which compresses the weightings of the model into 1-bit for each scalar. Arguably this

research has been seen before in the paper by Wang et al. [19] but its application to Federated Learning seems promising.

2.4.1 Statistical Distributions of Data

Subsequently after the original paper by McMahan et al., it was clear that the decentralised collaborative environment of Federated Learning would cause differences in the statistical distributions of the data, and thus formal definitions were created based on the statistical distribution of the data owned by the local clients, with this, three distinct Federate Learning approaches were defined (Figure 5).

2.4.1.1 Horizontal Federated Learning Horizontal Federated Learning is arguably the simplest from and uses a data distribution in which the data is distributed in an independent and identical fashion. This means that the sample space and label space is identical across the clients, and simply the actual data samples differ from client-to-client.

An important paper by Yang et al. [20] summarises the state-of-art up until the point in a formal fashion, and creates an important set of definition of the distributions discussed by researchers up until this point. Yang et al. defines Horizontal Federated Learning as follows:

$$\mathcal{X}_i = \mathcal{X}_j, \;\; \mathcal{Y}_i = \mathcal{Y}_j, \;\; I_i \neq I_j, \;\; \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

Figure 2: Horizontal Federated Learning - Yang et al. [20]

In (Figure 2), it is shown that Horizontal Federated Learning shares the same Sample space as other clients (denoted as X), and the same Label space (denoted as Y) but the actual samples are different (denoted as I). An example of this would be two local hospitals that have different patients, but the same type of data is held about all the patients (e.g. Name, DoB, address, blood type etc.)

2.4.1.2 Vertical Federated Learning Vertical Federated Learning refers to the approach required to perform aggregation when the distribution of data across clients in a non-independent and identical fashion (non-IID). McMahan et al. [15] were the first to conduct the testing of Federated Learning on non-IID data, and shortly after by Hardy et al. [7]. Both McMahan et al. and Hardy et al. refer to non-IID being a more complex statistical distribution, but extremely wide-spread in the wild.

In (Figure 3), it is shown that Vertical Federated Learning may not share the same Sample space as other clients (denoted as X), and may not share the same Label space (denoted as Y) but the actual samples are common across multiple clients (denoted as I).

$$\mathcal{X}_i \neq \mathcal{X}_j, \;\; \mathcal{Y}_i \neq \mathcal{Y}_j, \;\; I_i = I_j \;\; \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

Figure 3: Vertical Federated Learning - Yang et al. [20]

An example of this Vertical statistical distribution would be two online retail stores featuring common customers, but each store holds different types of data for each customer (e.g. storeX holds Name, Address, Shoe Size and StoreY holds Name, DoB, Shirt Size).

2.4.1.3 Federated Transfer Learning Federated Transfer Learning (FTL) is the condition in which Horizontal and Vertical distributions are not compatible as there are a few common data samples or labels amongst the clients or the dataset is very small in general. The few common data samples and/labels are used to build a sub-model that predict and generate data samples for predicted data labels. This is effectively a form of extrapolation in which a small amount of data is extrapolated to a point in which Horizontal or Vertical Federated Learning can be conducted.

Federated Transfer Learning uses a concept named "Transfer Learning" that was defined for general machine learning, but was adapted by and introduced with the federated environment by Liu et al. [13]. Transfer learning uses generative machine learning sub-models to generate an output of the features of the limited datasets, and then these features ares compared for similarities to generate a sample/label space with common features. It is evident that building sub-models to generate a larger model is computationally expensive however in the limited case of a small and non-common dataset, it may be necessary to use.

$$\mathcal{X}_i \neq \mathcal{X}_j, \;\; \mathcal{Y}_i \neq \mathcal{Y}_j, \;\; I_i \neq I_j \;\; \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

Figure 4: Federated Transfer Learning - Yang et al. [20]

In (Figure 4), it is shown that Vertical Federated Learning may not share the same Sample space as other clients (denoted as X), and may not share the same Label space (denoted as Y) and there may be no common data sample across the clients (denoted as I).

An example of this statistical distribtion would be two sets of retail businesses that operate in completely different sectors, but may share an extremely small sub-set of similar customers. (e.g. A bank hold customer mortgage information, and a furniture store features the same customer purchasing furniture for their new home).

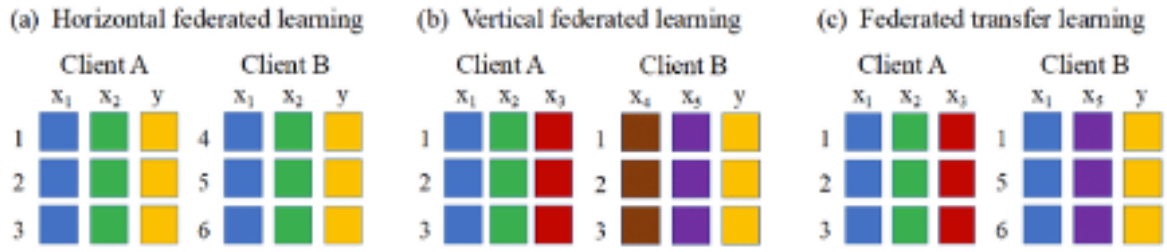

Figure 5: Data Distributions - [4]

2.5 General Challenges of Federated Learning

Given Federated Learning is a new and emerging technology being researched and adopted by large technology companies like Google, most of the research has been about the improvements and testing of the three learning approaches. However, an important collation paper by Li et al. [11] was to shift the discussion from use-cases and implementation types to risks and challenges.

This paper defines four distinct problems with the concept of Federated Learning and the implementations originally proposed by McMahan et al.[15] as follows:

1. **Communication Problems:** Whilst conducting centralised machine learning would require the transmission of the full dataset, and it is clear that Federated Learning updates would be much smaller in comparison, but Li et al. identified that having a large number of participating clients sending frequent model updates with extremely small changes is also inefficient. Local iterations of model updates before contributing the model update to the global model seems to be a promising direction to follow, as proposed by McMahan et al. [15] in the original paper but this still remains a problem for further testing.

2. **System Heterogeneity Problem:** This is a problem that is directly caused by the concept of Federated Learning where any device in the wild can contribute to the model, whether a laptop, mobile or IoT device for example. The "System Heterogeneity Problem" is concerned with the range of hardware, network, software and power resources of the participating clients and how this should be irrelevant to the model but currently causes some adverse effects in the wild. Li et al. are actively researching this problem and express that research is further required into this problem, with three possible solution avenues:

    - *Asynchronous Communication* can allow for clients with lower computational or network resources to still contribute to the global model given that the last time the client pulled the global model was within an acceptable time range.

- *Active Sampling* can arguably be considered as a form of favouritism where clients that are attractive in terms of computational and network resources are selected to participate in the update rounds more frequently, but this can introduce bias into the model given that the same data source is repeatedly selected and thus the global model will skew towards the data of the attractive clients and this may not be representative of a majority of clients.
- *Fault Tolerance* allows the aggregation server to ignore updates of failing devices, with the risk of bias similar to Active Sampling, or by introducing failover redundant clients that are linked to high-risk clients that increase the likelihood that another client can contribute to the participation rounds if the main client fails. This then favours more resourceful clients and skew the models in their data's respective direction.

3. **Statistical Heterogeneity Problem:** This is the problem discussed initially by McMahan et al. in which the Federated Learning system should accept both IID and non-IID data distributions which is handled through the respective Horizontal and Vertical Federated Learning. However Li et al. states that additional data distributions can be created if Federated Learning allowed for learning from peers rather than a centralised aggregation server or if there existed more than one server (e.g. a tiered approach of aggregating servers), and therefore this poses more complexity that requires further investigation.

4. **Security/Privacy Problem:** Arguably the most important problem mentioned by Li et al. and an active research avenue of Machine Learning in general. Security and Privacy attacks exist on current Machine Learning models and the effect of these attacks in a Federated setting needs to be tested thoroughly. The threat of model manipulation and data leakage is real, and common fixes to these have been applied to centralised Machine Learning, but these must be tested for applicability in a Federated setting too.

2.6 Security and Privacy Challenges of Federated Learning

With papers from Li et al. indicating that Security and Privacy of Federated Learning must be analysed, this evidently kicked off an explosion of research, with the most obvious being to test if current centralised machine learning security and privacy attacks are applicable in a Federated setting, and most importantly, whether they posed a viable threat to the system.

Lyu, Yu, and Yang [14] published one of the first papers that defines the current threat landscape in a general fashion and gives prospective implementers of Federated Learning

an extremely useful insight into the threats posed by attackers and their respective attack channels.

Lyu et al. breaks down the security and privacy threats to federated learning using a holistic threat model assessment, in addition to a follow up paper by Enthoven and Al-Ars [6] as follows:

- **Attack Locations:**
  - An *Insider Attack* is defined as a client that is participating in the federated learning process, or even the aggregation server itself. Any client within the defined architecture model is assumed to be malicious, and therefore can possibly launch an insider attack. A single malicious client that attempts to disrupt the system or invalidate the model is considered a "Single attack", whereas a stronger and more organised group of clients with the same malicious objective and a larger attack vector can be considered a "Sybil Attack". If the group of malicious clients have an additional aim of detection avoidance, this can be considered the strongest from of insider attack, and this can be done using a "Byzantine Attack" in which the malicious clients act honest and mimic expected model updates.
  - *Outsider Attacks* are considered malicious actors not defined in the federated learning architecture, such as eavesdroppers or malicious users that use the global model in an unaccepted way upon deployment such as Google Cloud AI users.
- **Attack Goal and Respective Phases:** With any machine learning system, there are two distinct phases in which an attack can be deployed based on:
  - If the goal of an adversary is to force the model to enact an unexpected behaviour in a random or designed fashion this can be conducted in the first phase, or corrupt the model. The first phase is considered the *Training Phase* in which the model is being constructed from training data and the system is in an iterative learning stage in which the accuracy is attempting to converge as high as possible.
  - If the goal of an adversary is to view/infer data about a victim that the attacker is not meant to be aware of then this can be conducted in the second phase. The second phase can be considered as the *Testing Phase* in which the model has already been constructed and a client can input test data to use the model. At this phase attacks are loosely considered privacy attacks in which an attacker attempts to learn information about the underlying model, or the training data that was used to build the model in the training phase.

- **Adversarial Initiative:**
  - An adversary can launch an *Active Attack* in which the parameters and inputs of the model are manipulated to the adversaries requirements. These attacks can be used to perform adversarial tasks such as forcing a model to misclassify data, or to have sporadic random outputs such that the system is dysfunctional. These attacks can be detected using simple tools such as anomaly detection
  - *Passive Attacks* require little to no input from the adversary and are therefore much harder to detect as there is no trace or anomaly to detect.

2.7 Security and Privacy Attacks on Federated Learning

2.7.1 Training Phase Attacks

Training phase attacks are generally considered forms of security attacks in which either data or the model is poisoned with the objective of reducing the accuracy of the model or introducing a backdoor in the model. As presented by Lyu, Yu, and Yang [14] and observed by Cao et al. [3], the two poisoning attacks that can take place are defined as follows:

- **Data Poisoning:** In machine learning, the training dataset is used to build the model, and evidently [14], if the training dataset is poisoned, then the model that is built will not function in the expected way. For example, an adversary can either attack the training data and change the data such as the trained model output is different to the original, or manipulate the test data such that the model is not able to classify the input samples. There are two conditions presented by Lyu et al. to this attack depending on the access provided to the participating clients:
  - A *Clean Attack* is the condition in which an adversary cannot directly manipulate the data in the training dataset as there may be some authentication checks on the dataset to detect manipulation.
  - A *Dirty Attack* is the condition in which an adversary can directly manipulate the data in the training dataset and can alter existing data (e.g. a label flipping attack where data samples are change from TRUE to FALSE) or by introducing new data samples that skew the dataset distribution. Additionally, an adversary can leverage both to introduce a backdoor such that a specific sub-set of training data is manipulate to force a missclassification backdoor in the model.
- **Model Poisoning:** In Federated Learning, the model updates are communicated to the aggregation server in the form of weights, and this provides an attack opportunity to an adversary. In the moment after the weights have been computed

and before the local model updates are transmitted to the aggregation server, an adversary has the opportunity to poison the weights of the local model and thus by design the global model. This paper believes this is because there is a strong assumption that the communication of the clients and the server is secure, and any attacks on the data would be considered data poisoning attacks as proposed by Hardy et al.[7] in which they showed that homomorphic encryption should be used when transmitting the updates beyond the local boundaries, and Federated Learning should always assume a secure communication channel.

The corruption attacks have ranging applications. The goal may be that the model is a target and the adversary just wants to make it unusable for denial-of-service purposes, and the output of the model can be random, or an adversary wants to inject a backdoor into the model such that a very specific input leads to a missclassfication in the favour of an adversary, for example this could be to force a model to accept forged documents.

2.7.2 Testing Phase Attacks

Testing phase attacks generally have an adversarial objective of learning information that should not be accessible by an attacker, and thus can be generally considered as privacy attacks. This can be done by maliciously choosing certain inputs and observing the output of the model to deduce information about the model, training data or other participants.

The attackers access to the model can be defined using the standard *white-box access* – where the adversary is aware of the structure of the model internal workings, participants and aggregation server – or the opposite *black-box access* – where the attacker is unaware of the model and only has query access to define inputs and observe outputs. Based on this, testing phase attacks can be summarised into the following five categories of privacy related attacks:

- **Class Representative Inference Attacks:** An adversary attempts to generate samples that are statistically similar to the data samples that exist in the dataset, which allows the adversary to learn further information about the original dataset.
- **Membership Inference Attacks:** An adversary attempts to learn if a sample was a member of the training dataset as defined by Shokri et al. [17]. An interesting example of this is in the medical field, where it may be possible for "an attacker can infer whether a specific patient profile was used to train a classifier associated with a disease". [17]
- **Property Inference Attacks:** This is a higher-level attack that looks the aggregation of the data samples, and the adversary attempts to learn general characteristics

of the original dataset such as the distribution and certain features of the data. This is a less common attack as it extends the attack vector beyond the actual dataset, and looks at the properties of the dataset instead.

- **Sample/Label Inference Attacks:** This is a full reconstruction attack where an adversary attempts to rebuild the original dataset samples, labels or both. This is different to class representative inference, as with this attack the adversarial goal is to rebuild the same data rather than data that has a similar distribution or aggregate properties.
- **Model Inversion Attack:** Arguably The most common form of an information Inference attacks is the Model Inversion Attack – where an adversary iteratively provides inputs to a black-box model and observes the output (like an oracle) and from this they can essentially try all the combinations of inputs that reflect a desired output and thus revealing the input data. Another attack could be the use of Generative Adversarial Networks (GAN). GANs are a form of generative models, and are commonly used to generate data with a similar statistical distribution with similar class representatives and data properties for training good models where a lack of data is available.

2.7.3 Backdoor Attacks

A backdoor attack is an interesting focus as it is designed using a hybrid combination of different security attacks to introduce a function into the model that it was not initially designed to do.

Bagdasaryan et al. [1] initiated research into the applicability of backdoor attacks on Federated Learning systems by following on from previous research and publishing an immensely beneficial body of work. The main driver for this research is that Bagdasaryan et al. immediately identified that there were no checks in place to ensure that local models updates had been performed correctly, and the only mentioned possible checks were anomaly detection as per the paper by Bhagoji et al. [2] in which it was stated that the only metrics an aggregating server can check is the similarity of the malicious update to the other honest updates received.

Therefore, it was proposed that if a backdoor could be introduced, and the statistical distribution of the model remained similar, and the performance (i.e. accuracy convergence and loss) of the model also remained similar, it would be clear that the aggregation server would accept a malicious model update.

Within the objective of introducing a backdoor, Bagdasaryan et al. defines two types of methods an adversary can choose

- First, a **Semantic Backdoor** is defined as the ability for an adversary to introduce a backdoor based on features of the data that are naturally occurring. An example of this would be if an image dataset had pictures of cars, a naturally occurring feature would be the colour of the car. An adversary can then choose a naturally occurring feature (e.g. purple cars) and poison the labels of these data samples such that any image that contains this feature is misclassified. This does not affect the accuracy or performance of the model, it just means that the model behaves incorrectly on a certain condition. This attack is essentially a passive form of attack as the data samples themselves are not modified and I would argue this is a weaker form of attack as the adversarial advantage required to know all the features of the dataset is high, in addition to the possibility of introducing a backdoor that could be triggered by another participant since other participants will have data that has similar features to the attackers dataset.

- Second, an arguably stronger form of backdoor is a more active attack in which a feature is injected into the images such that when this artificial feature is detected, the output will be an engineered misclassification. A common form of this is the **pixel-pattern backdoor** attack where an adversary injects a specific combination of pixels into an image that does not occur naturally, and changes the label for this image. Therefore, once the model is trained, any image with this combination of pixels will be misclassified with a high confidence. This attack requires more contribution from the attacker as both the training samples and the labels must be manipulated, and then to confirm the existence of the backdoor, some training samples must be manipulated too. This type of attack was proven to be extremely effective by Bagdasaryan et al. in addition to a follow up paper by Sun et al. [18] in which it was found that a pixel-pattern backdoor can keep the same accuracy convergence as the original model, with near 100% accuracy on the missclassfiication sub-task, and concluding that this may be one of the strongest attacks on Federated Learning.

2.8 Evaluation and Direction

This survey of the current literature within the scope of Federated Learning clearly shows the current problems in this new system. Following on from this survey, this paper will focus *only* on security problems within Federated Learning due to the scope of all attacks and other problems aforementioned.

3 Methodology

This paper's core high-level objective is a systematic comparison of performance and security attacks on the Federated Learning system, and thus as the Federated Learning systems comprise of three distinct statistical distributions that dictate the operating principles, it is worth nothing that this paper will *only* focus on Horizontal and Vertical Federated Learning due to the fact that these are the most common and additionally the only direct systems, in comparison to Federated Transfer Learning that requires pre-processing to generate and extrapolate the limited samples to be computed before building the full federated system.

3.1 Hardware and Software Limitations

Most, if not all, the papers developed within the machine learning research sphere tend to value the deployment of the models on cloud infrastructure, given the recent popularity of machine-learning-as-a-service. It is not clear whether McMahan et al. utilised cloud computation for the testing, but given the paper was developed at Google it is highly likely this was deployed on a Google Cloud Platform cluster.

The experiments conducted by this paper do not utilise the cloud and are instead developed on a local machine with an Intel 2.6GHz Quad-Core i7 CPU, 16GB of DDR3 RAM and a dedicated Nvidia GeForce GT 750M 2GB graphics card. This machine is not comparable to running computations on the cloud, however the dedicated GPU was utilised to optimise performance given the limited resources.

The experiments were developed in Python, utilising common statistics and machine learning libraries such as NumPy and PyTorch.

3.2 Dataset Selection

Dataset selection is key to dictating the model built and therefore the attack goals. The simplicity in the dataset and thus a model that would not be computationally expensive was key to the selection, given the objectives of the experiments and the limited computational resources. The simplest form of supervised machine learning models are usually classifiers, in which an input (image, text, sound etc.) is paired with a label, and passed to the model for training.

This paper selected the MNIST dataset, which is a set of 60,000 images made up of 28x28 pixels that contain handwritten digits from $0 - 9$ and their associated labels, in addition to 10,000 more images for testing. It is worth noting that should a model require more images, there is an extended dataset with 240,000 images in the EMNIST dataset,

however for this use case the core MNIST dataset was overly sufficient.

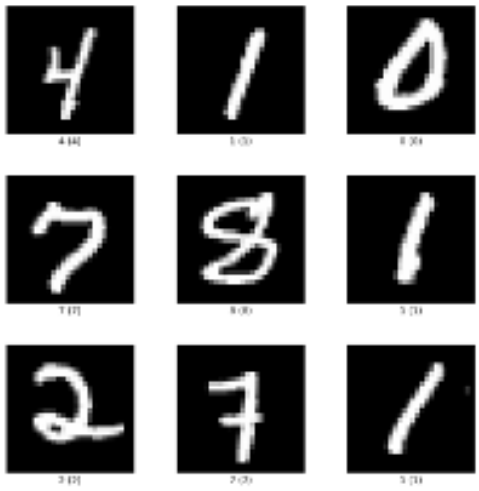

Figure 6: MNIST Dataset Sample

3.3 Model Objectives

As a simple classifier, the supervised model has a single objective of predicting the label of an unseen sample with high accuracy based on trends learnt from previously seen similar data. This is effectively sorting new data into classifications based on the classifications of similar data.

In this instance, if the input to the model is (figure 7), the desired output should be the number the system believes most closely represents the input image, and an accuracy percentage. In this example, the output would be expected to be the number "8".

Figure 7: MNIST Dataset Sample of Number 8

3.4 Model Architecture

The selected architecture is the simplest Feed-forward Neural Network, more complexly known as a Multi-Layer Perceptron. This allows us to define the neural network as an input layer and an output layer. The weightings from each node to the next are adjusted after every sample-label pair is processed by the system in a back-propagation process as follows:

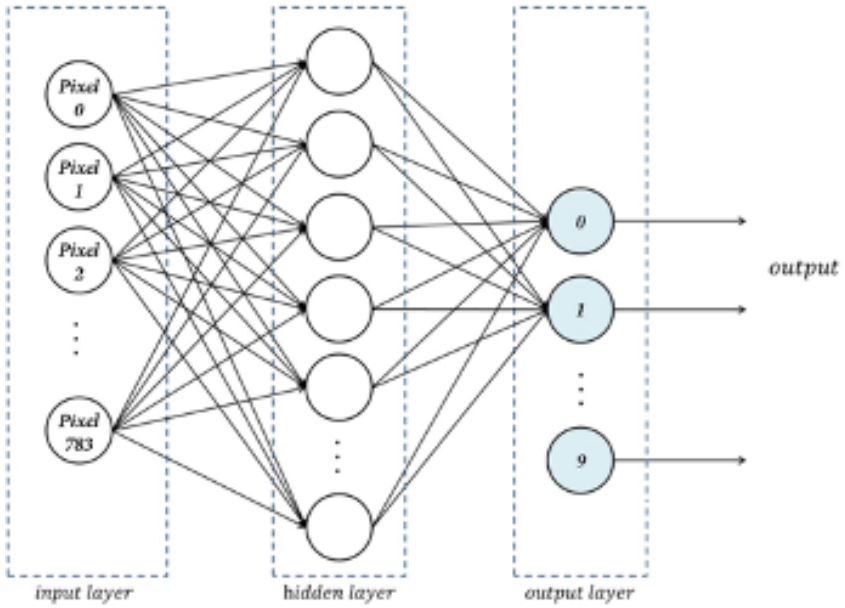

Figure 8: Neural Network Architecture Layers

In this design (demonstrated in figure 8) the pixels of each image are loaded through the input layer sequentially (Px 0,1,2...783), then the machine processes the pixels and tries to determine the written number in the image through a series of hidden layers, then in the output layer there are only 10 outputs defined (0 – 9) and associated accuracy for each respective number. The more samples and labels that go through the network, the higher the accuracy. [Figure 8]

The reason this paper did not opt for a more complex model architecture such as a convolutional networks is due to the simplicity of the classifier. The input types are fixed (i.e. an image of 28px x 28px with a number from 0-9 drawn) and the learning is supervised with complete sample-label pairs, and therefore anything more complex might yield negligible accuracy gains and exponentially increase computational resources required which were not available in the case of these experiments. Additionally, the objective of this paper is not to optimise the machine learning algorithm, but to experiment and compare the behaviour of federated learning as whole under certain circumstances, and thus the simplest model architecture that would a high accuracy of 90%+ would be acceptable.

3.5 Federated Learning System Flow

Following the design for both Horizontal and Vertical Federated Learning defined by [15], [9], and [8]. This is the overall flow expected for both Horizontal and Vertical Federated Learning systems.

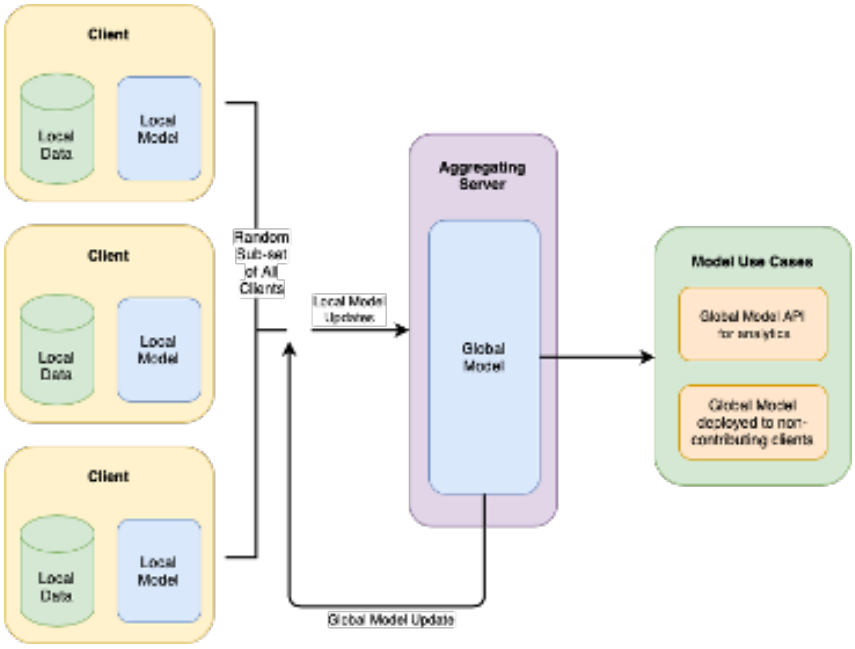

Figure 9: Federated Learning System Flow

Since this is a simulation of federated learning and clients are created virtually, there exists no "on-device data" until a pre-processing occurs to distribute the data to each client. The 60,000 sample-label pairs are distributed to all the 100 clients as follows:

- **Horizontal Federated Learning Setup:** Given the identical distribution, the horizontal federated learning simply consists of dividing the total sample-label pairs amongst all the clients.
- **Vertical Federated Learning Setup:** Given the non-identical distribution, the vertical federated learning uses a system of sharding, in which all the image labels are sorted (i.e. 000...,111...,222...,...999...) and then all the images are divided into sets of 200 shards of size 300. Then the 100 clients are allocated 2 shards each.

3.5.1 Aggregation Algorithm

Averaging the local model updates on the aggregation server can be done in a number set of ways. To keep the testing as baseline as possible, the defined algorithm for the aggregation of the updates will be Federated Averaging (FedAvg) designed by McMahan et al. [15] defined below in figure 10:

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figure 10: Federated Averaging - McMahan et al. [15]

Federated Averaging works with four key parameters represented in the algorithm in figure 10. The full number of clients is defined as K, and C is the percentage of clients (i.e. client sub-set of all clients) that are selected to participate in each round. E is the number of local training rounds, and B is the minibatch size, which is defined as the full dataset for the purposes of FedAvg.

3.6 Threat Model

An adversary is given white-box access to participate in the Federated Learning system. The adversary may control one (Single Attacker) or multiple contributing clients (Sybil Attack), however, by Federated Learning design, not all clients may be selected for participation.

The malicious clients are then able to manipulate their own datasets and/or local model updates such that one of the following objectives is met:

- Manipulate the local model randomly for one or more malicious clients that are actively owned by an adversaries such that the global model accuracy is reduced.
- Engineer the local model of one or more malicious clients such that the global model accuracy is not reduced and a backdoor misclassification exists for exploitation.

3.7 Experiment Outline

The experiments that will be conducted as part of this paper are generally grouped into performance experiments and security attack experiments as follows:

3.7.1 Performance Experiments

In order to gain an understanding of the two Federated Learning systems, a baseline system is built for both Horizontal and Vertical Federated Learning with the parameters tested initially based on the paper by [15] and [9] then two further experiments are run such that the parameters are adapted to suit current computational resource limits.

Using the defined parameters in the original paper published by McMahan et al. as follows, a baseline experiment is created to reveal a performance benchmark on which the results of the attacks can be compared;

- Global Training Rounds = 10,
- Total Number of clients = 100,
- Percentage of participating clients per round = 10%,
- Local training rounds before transmission = 5,
- Local Batch size = 10.

Given that the objective of this paper is to assess the effect and relevance of various security attacks, the performance of the baseline model should be *"good enough"*. This is deemed as 90%+ and thus the baseline parameters are then incremented such that the model converges at the highest "good enough" accuracy given the limited computational resources, and also a time balance. For example, given 95% accuracy in 20 global training rounds that takes 2 minutes to run, vs 96% accuracy in 50 global training rounds that takes 10 minutes to run, the "good enough" policy dictates that the first set of parameters are used.

Following the selection of the baseline parameters, global training rounds and local training rounds are selected as independent parameters as they have the most effect on accuracy within a certain time period (e.g. 2 minutes), and these rounds are tested to select the best choice for the baseline model:

1. Experiment: Iteratively change the value of the **global training rounds** from 10 - 50 to find the "good enough" value.
2. Experiment: Iteratively change the value of the **local training rounds** from 2 - 10 to find the "good enough" value.

3.7.2 Security Attack Experiments

3.7.2.1 Data Poisoning As mentioned previously, data poisoning attacks have varying adversarial objectives, and the objectives of these experiment is to observe the effect of randomised data poisoning of both the training and the testing datasets with the adversarial goal of either reducing the accuracy of the global model possibly to a point of denial-of-service, or cause the model to behave differently.

In order to conduct these experiments, this paper introduces a new method of defining **"Attack Strength"** and implements corresponding parameters to represent this:

1. The first parameter is the **"spread"** of the attack, and is defined as the percentage of samples that are selected to be poisoned in the dataset.

2. The second parameter is the **"potency"** of the attack on those samples (i.e. how many pixels are changed in the image is selected to be poisoned).

The Attack Strength of the attack can be defined as follows:

$$\textbf{\textit{Attack Strength = Spread x Potency}}$$

Whilst the attack spread is difficult to present as it is simply a parameter, the attack potency is easier to visualise. The (figure 10) shows a hypothesis of what an attack of varying potency would look like starting with the benign image sample, and increasing the attack to an arbitrary value.

The increase in poisoned pixels reduces the confidence of the model, and this causes the average accuracy to decrease on each image, therefore in aggregate, it is clear to hypothesis that the higher the spread of the attack, with a stronger potency, should yield a stronger attack and thus reduced overall model accuracy.

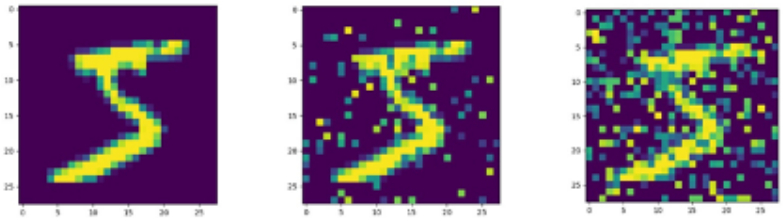

Figure 11: Attack Potency

3.7.2.2 Training Data Poisoning The training data is arguably the source of the model, and accuracy is decided based on these inputs. Therefore, the experiment is to observe the possible accuracy loss and convergence interruption caused by poisoned data samples as follows:

- **Low Strength Attack:** Fix the attack potency to 20% and increment attack spread to see how much of the data needs to be poisoned to see a drastic loss of performance on both Horizontal and Vertical Federated Learning.
- **Medium Strength Attack:** Increment the attack potency to 40% and increment the attack spread to see how much of the data needs to be poisoned to see a drastic loss of performance on both Horizontal and Vertical Federated Learning, in addition to how much impact an increase of 20% potency on performance.

3.7.2.3 Test Data Poisoning Similar to training data, should the test data be poisoned, then the model cannot be used correctly and thus quantification of this effect is relevant to understand the strength and relevance of this attack.

The same two parameters for "spread" and "potency" are used as per the training data attack, however as a hypothesis, manipulating the test data is likely to be less effective with a low spread, and thus the tests start with a higher amount of samples being selected for poisoning – 50% and then increments to a higher percentage.

3.7.2.4 Label Poisoning An interesting experiment on the effect of data poisoning is to observe the effect of poisoning labels instead of samples (e.g. image of number 8 label changed to 4). The spread of the attack is incremented in steps of 10%, and the potency of the sample-label pair is 40% but now the label is changed instead of the selected sample being manipulated, it is the label instead. The value of the selected labels are changed to a random number from 0 – 9, and this test is conducted on both Horizontal and Vertical Federated Learning systems.

Note: It is worth noting that this experiment is only applied to training data as the hypothesis is that a label poisoning attack on test labels would yield an illusion of an impact. For example, if the model learns that an image of the number 8 is labelled as 8, and then the test image is an image of the number 8 with the label 6, then the test accuracy of the model is reduced when in reality the model is operating correctly as an image of a number 8 is not classified as a 6.

3.7.2.5 Backdoor Attack The objective of this attack is to allow an adversary to inject a pixel-pattern into any images and have the model classify the image as the adversary requires, regardless of the image content. Previously, this paper introduced the

concept of a backdoor using a pixel-pattern attack as having the following attributes:

- Does not reduce overall accuracy of local or global model.
- Injected pixel-pattern should not be detectable by machine or human eye.
- Backdoor can never be initiated unless a specific combination is present that will never occur naturally or accidentally.

Therefore, for this experiment, any image with the pixel-pattern of the first two rows as black will be set to misclassify as a Zero (Figure 11) where [A] is the original sample, and [B] is the modified sample with the injected pixel-pattern. ***Note:*** This pattern was chosen as it is not immediately distinguishable by humans if the images were not side by side, and simply looks like a thick border rather than an obvious pixel combination, however, the adversary can choose any pixel combination they desire.

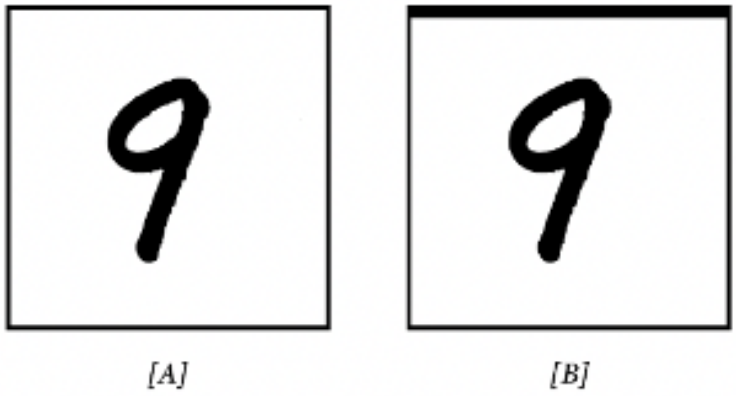

Figure 12: Pixel-Pattern Injection

This design is conceptually the most beneficial for an adversary but in theory must be balanced, so it is worth noting the following conditions during experimentation:

- If the number of images on the local client that have the pixel-pattern is not high enough, the model may not learn the features of the pixel-pattern and associate it as a feature of the classification.
- If the number of images on the local client that have the pixel-pattern is too high, the model will not learn any features of the benign dataset, and thus all images will be misclassified as the pixel-pattern becomes too dominant.
- If the pixel-pattern itself is not distinguishable enough, e.g. a random singular pixel in the middle of the image, the model may have correlating this feature to a classification if at all correctly learning the misclassification as required by the adversary.

Therefore, given the accuracy is expected to remain similar, two distinct experiments are conducted to verify the existence and correct function of the backdoor:

1. Once the model has reached a high accuracy and convergence, the test data to be used must include some benign samples in addition samples that also have the pixel-pattern. If both benign and backdoor samples are correctly classified, then the existence of the backdoor is confirmed.

2. Additionally, another experiment is to use test samples that have been injected with the pixel-pattern. This is expected to reduce the testing accuracy to near zero, whilst maintaining high backdoor test accuracy.

It is worth noting that the potency in this form of attack is not amount of pixels changed, but the number of clients introducing the backdoor, thus the strength of this attack will depend on two distinct variables:

- The number of clients manipulating their individual models such that the backdoor is trained. If the number is more than 1 client, then this is considered a Byzantine Attack.

- The number of images with the pixel-pattern injection in the dataset.

3.7.2.6 Model Poisoning The adversarial objective of the model poisoning attack is to reduce the accuracy of the global model, or completely cause denial-of-service, however the adversary in this case does not have access to the dataset to manipulate it. Therefore, this attack vector occurs at the moment before the local model update is transmitted to the aggregation server, and the weightings of the local model update are manipulated.

There are two distinct scenarios which must be evaluated for both Horizontal and Vertical Federated Learning as follows:

- Random Attack: The independent variable in this experiment is the number of the clients the adversary owns, which is a pre-defined parameter that is allocated as the data is being split vertically/horizontally.

  Note: It is worth noting that an adversary may own 10 clients connected to the system, but since the clients participating in a update round are random, not all of the malicious clients are selected are selected to participate.

  This attack is a representation of an attack in which the client device is compromised (e.g. through phishing and the adversary indirectly owns the clients, and therefore it is more random) and the adversary uses these honest-but-compromised clients to manipulate the global model. This attack is a clear form of **Sybil Attack** and thus must be evaluated.

- Targeted Attack: This is opposite to the random attack, in which an adversary owns the clients and joins the system with the intention of disruption, and is now always chosen in the rounds as the same client is fixed. For example, if client number 77 is malicious, then client 77 is chosen to participate, and the model update is guaranteed to be malicious.

For both of these experiments, any model update that is maliciously manipulated is considered to be random data with no meaning, and thus the adversarial goal is simply to reduce overall accuracy or possibly perform denial-of-service depending on the strength of the attack, which will be the independent variable of this experiment.

Therefore, for both random and targeted model poisoning attacks, this paper introduces a selection variable - A percentage variable is introduced to define which clients are malicious in the pre-processing stage. This percentage variable will dictate the strength of the attack, as it is hypothesised that the more malicious clients participating in a round (and thus a stronger Sybil Attack), the higher the attack strength.

4 Experiments: Results and Analysis

Following the defined testing strategy and objectives defined in the methodology, the following section represents the results of the experiments and brief analysis of each experiment:

4.1 Performance Experiments

McMahan et al's design of federated learning included performance results using parameter's the team deemed suitable for their computational resources and time. This paper implements both the Horizontal and Vertical Federated Learning systems with said parameters as follows:

- Global Training Rounds = 10,
- Total Number of clients = 100,
- Percentage of participating clients per round = 10%,
- Local training rounds before transmission = 5,
- Local Batch size = 10.

Computing this experiment, the results are presented in Figure 13, which shows that the Horizontal Federated Learning system converges to over 90% accuracy within the first few rounds, whereas the Vertical Federated Learning system converges slower and reaches a peak accuracy of approximately 70%. These results are consistent with the results obtained in the original paper and additionally other papers which feature baseline results.

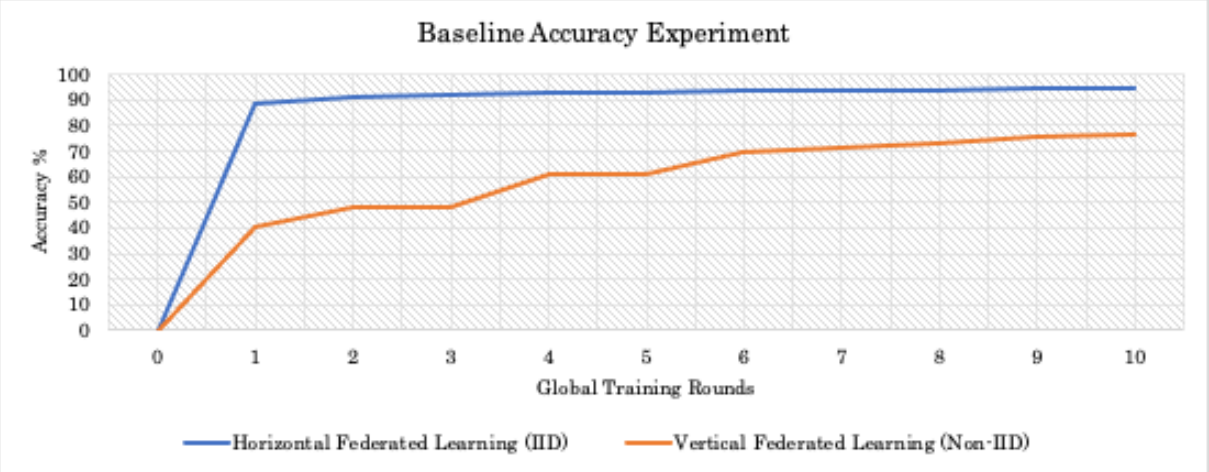

Figure 13

In order to conduct security attacks, this paper conducts some further experiments to optimise the parameters on the both Horizontal and Vertical Federated Learning systems, and to determine the parameters that will yield the highest accuracy in the least time possible based on the limited computation resources as outlined in the methodology section. The following results are of the results of these experiments:

4.1.1 Global Training Rounds

The default parameter of the number of global training rounds (i.e. the number of times the sub-set of clients transmit their local model updates - currently set at 10 by McMahan et al. [15]) set for all the security experiments would need to be the middle ground between the best-case accuracy and worst-case time. An experiment using the default 100 clients with a 10% participation rate is conducted, but iterating through 10, 20, 30, 40 and 50 global training rounds to observe the final average accuracy given the number of global training rounds:

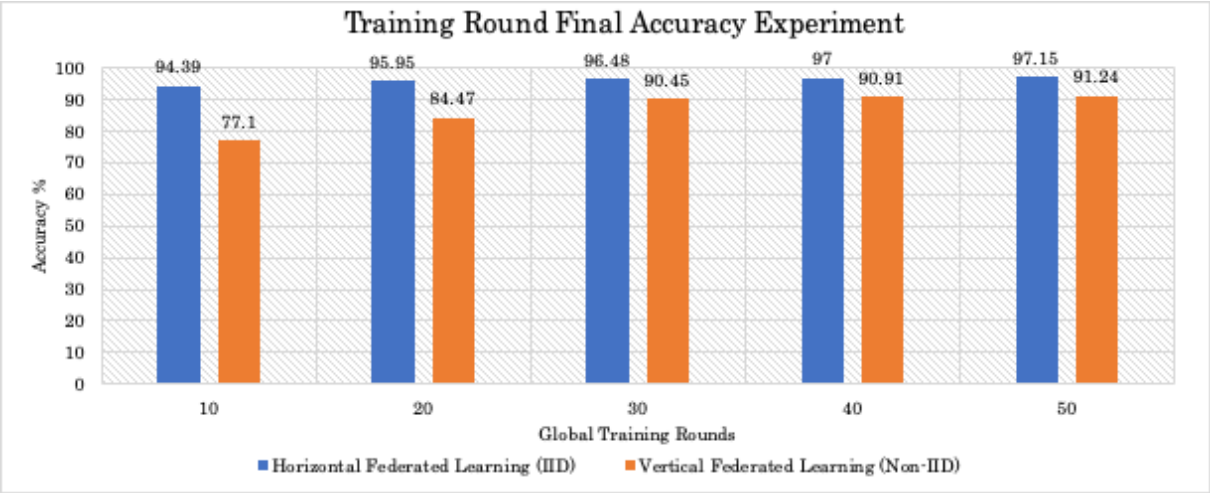

Figure 14

Figure 14 shows that this continues the trend in which Horizontal Federated Learning systems converge faster than Vertical, and reaches a higher peak, with Horizontal reaching over 90% accuracy in the first 10 rounds, whereas it takes Vertical Federated Learning up to 30 rounds to reach a 90% accuracy. The Vertical Federated Learning system continues to converge towards the 93% mark, however given limited computation resources, I would estimate that over 100 rounds would be needed to see over 95% accuracy which is extremely time consuming.

Therefore, this paper determined that **30 global training rounds** would yield over 90% accuracy for both Horizontal and Vertical Federated Learning Systems and therefore would be an optimal parameter for the forthcoming attacks.

4.1.2 Local Training Rounds

From the previous result, the parameter of 30 global training rounds is now fixed, and the experiment is conducted to determine the local training rounds required (i.e. how many local training rounds does each client conduct for every transmission of the model update to the aggregation server). The local training rounds are conducted locally and therefore the total amount of training would be defined as the number of global training rounds multiplied by the number of local training rounds, and thus choosing a number too high could exponentially increase computation time of each test. The results are of the experiments conducted for both Horizontal (Figure 15) and Vertical (Figure 16) Federated Learning systems are presented below:

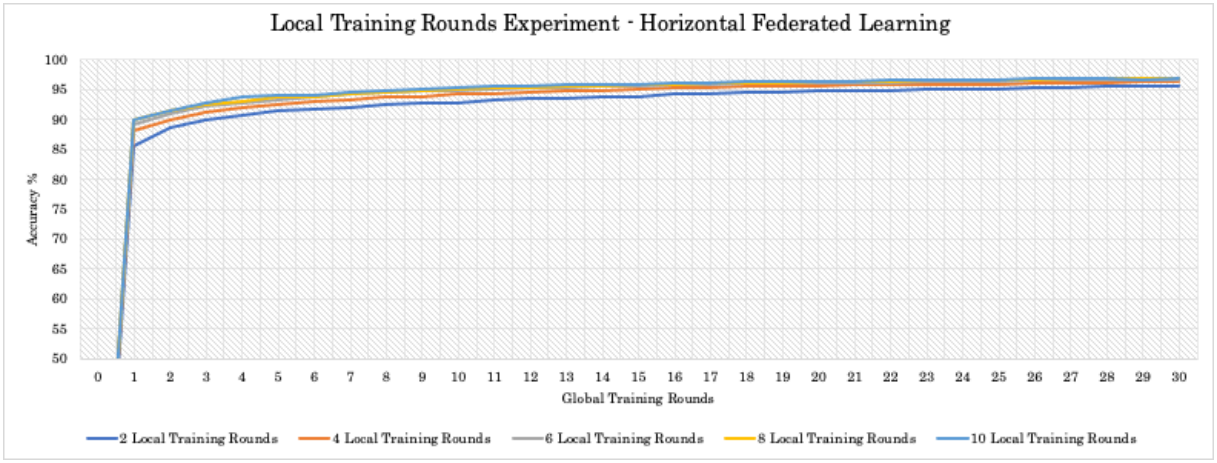

Figure 15

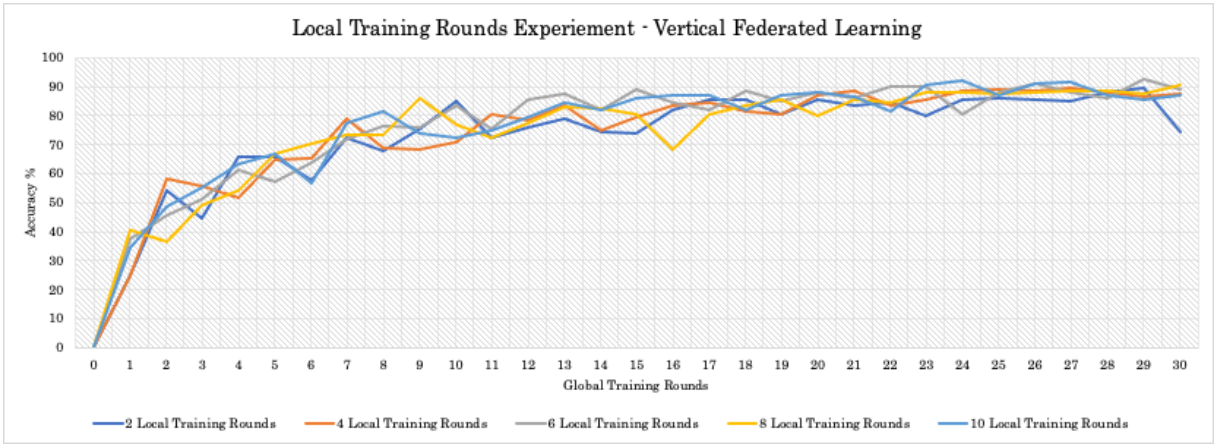

Figure 16

Whilst time is not displayed in these results, the more local training rounds per global training round typically results in a longer training time overall, and therefore a balance

must be struck between time and performance. The Horizontal Federated Learning system converges much faster than the Vertical Federated learning system and reaches high accuracy peaks. This is consistent with previous experiments and experiments conducted by other researchers. Additionally, the Horizontal Federated Learning system provides a consistent result, and the general trend is that the more local training rounds, the faster the accuracy is gained. For example, for 2 local training rounds per global training round, the accuracy reaches 90%+ after 4 rounds, however for 10 local training rounds per global training round, a 90%+ accuracy is reached on after only a single-shot in the first round.

The Vertical Federated Learning System is sporadic due to the unequal dataset distribution and therefore any gains made seem to be wiped away by low performing participating clients, but the convergence trend also seems to indicate that the higher the local training rounds per global training round, the system is more accurate in less rounds.

Therefore, striking a balance, this paper concludes that **6 local training rounds** would be sufficient for the experiments in addition to the 30 global training rounds giving a total of 180 training rounds per model. 6 local training rounds seems to start with a high accuracy and converges to a similar level as the high local training rounds, but takes much less time. Therefore, the middle ground between accuracy and time once again is the best option.

These experiments are necessary to create a benchmark to compare the results of the attacks, and therefore can see that the normal operating behaviour of a horizontal federated model is to converge at an accuracy of over 90% in the first 10 global training rounds, and then reach approximately 95-96% accuracy after the full 30 rounds. This is in comparison to the vertical federated learning system which takes a longer time to converge with 70-80% accuracy in the first 10 rounds and peaking around the 90% accuracy mark after the full 30 rounds. This will be the benchmark in order to assess the effectiveness of attacks in the following experiments.

4.2 Security Comparison

The performance experiments in the previous section have indicated the parameters to fix for the security attack experiments, and therefore as the methodology indicates, the attacks based on the follow up papers are implemented and evaluated as follows:

4.2.1 Data Poisoning Attacks

As mentioned in the methodology, there are two parameter's in these experiments that this paper contributes to effectively quantify the strength of the attack. The first parameter is "spread" of the attack, loosely the percentage of samples that are poisoned in the dataset,

and the second parameter is the "potency" of the attack on those samples (i.e. how many pixels are changed if the image is selected to be poisoned).

The following data poisoning experiments are conducted on both Horizontal and Vertical Federated Learning systems to test the strength of data poisoning given incrementing spread and potency, and applying it to the different data phases of the system – the testing phase and the training phase:

4.2.1.1 Training Data Poisoning In the training phase of the Federated Learning system, the Low Strength Attack and Medium Strength Attacks are conducted in order to test how much of the dataset needs to be poisoned (spread) and the poison potency required to reduce the accuracy. The first potency is fixed at 20% with the percentages incremented as per the test plan in the methodology:

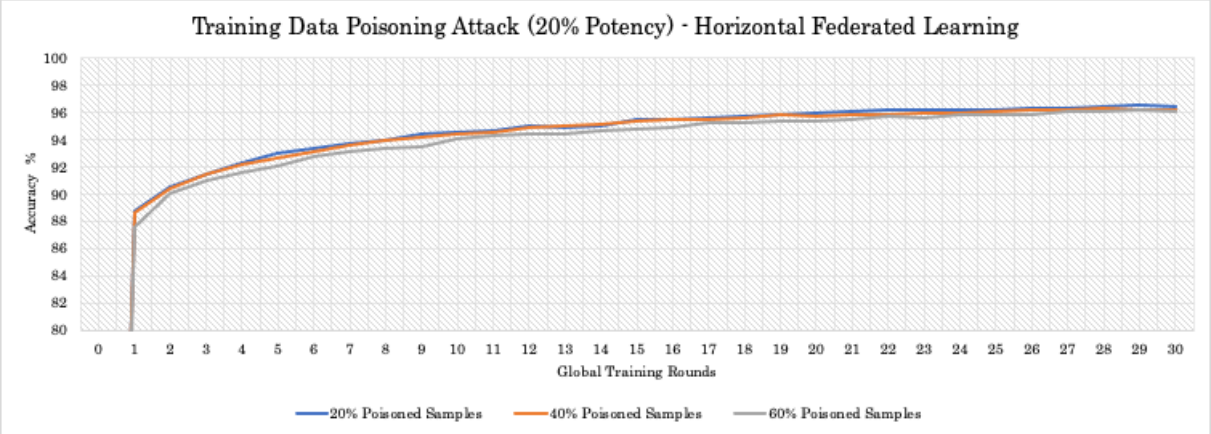

Figure 17

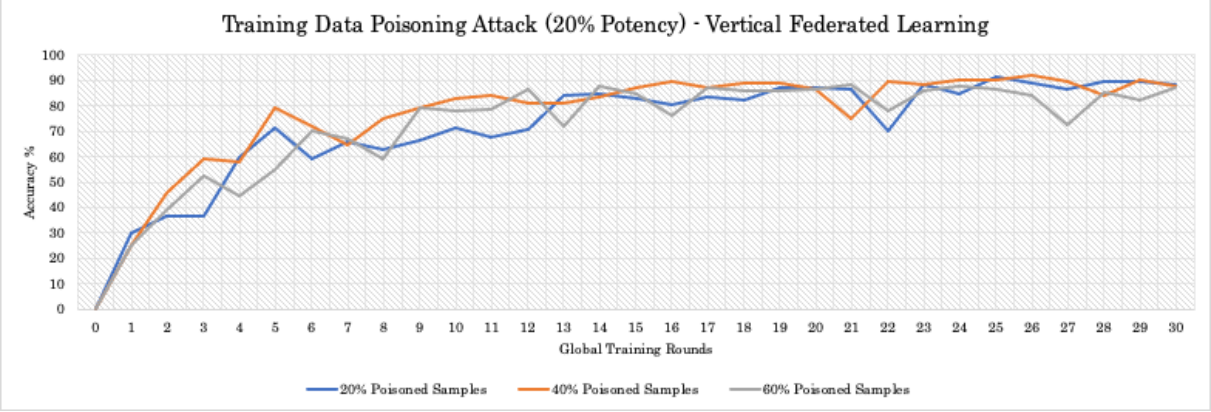

Figure 18

This Low Strength Attack experiment seems to have little-to-no impact on both the Horizontal (figure 17) and Vertical (figure 18) Federated Learning Systems. Both the

convergence and accuracy are relatively unimpeded and these results seem to closely reflect the results presented in the baseline experiments. The Horizontal Federated Learning system reaches the desired 90% accuracy in 2 rounds for all cases and then converges at nearly 96% after 15 rounds with minimal changes recorded across the rounds. This is also similar for the Vertical Federated Learning System in which 90% accuracy is reached within 15-20 rounds. It is clear that there is a minor effect on the accuracy from round-to-round with sudden dips of accuracy, but generally the performance is similar to the baseline system.

The potency is then increased to test its respective accuracy reduction and fixed at 40% with the percentages incremented:

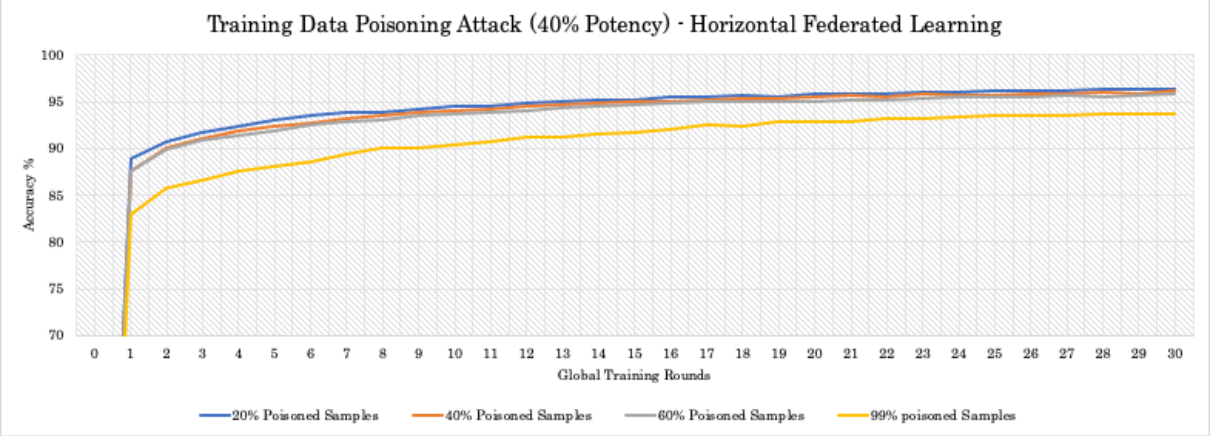

Figure 19

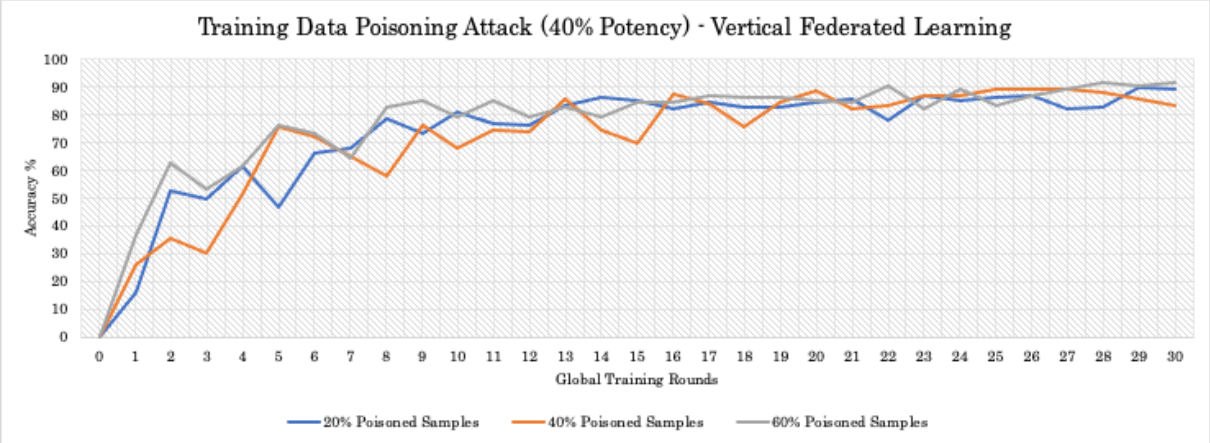

Figure 20

Once again, even with the increase in potency, the effect is negligible for most spreads. The Horizontal Federated Learning system (figure 19) seems to converge at 95% for all the percentages of data being poisoned with no interruptions or anomalous rounds present,

but with a potency of 40%, convergence takes more rounds at approximately 12-15 rounds to taper. While it's harder to judge the Vertical Federated Learning system (figure 20) in a similar manner, it is clear to see that the Vertical system has no trouble from converging at around the 15th round, with accuracy at nearly 90% on the 30th round. This is similar to the baseline results, with extra rounds required to gain momentum and converge at the peak accuracy.

In order to understand the true impact and why there was minimal changes in the results, an off-centre anomalous experiment of 99% was conducted on the Horizontal Federated Learning system (figure 19 - yellow line). The data samples were still being poisoned with 40% attack strength and this shockingly only reduced the accuracy by a negligible 5%. The model still converges at over 90% accuracy in a similar fashion to the other test cases even though 99% of the dataset is poisoned. The working theory for this lacklustre degradation of performance is related to the simplicity of the MNIST dataset, and this is discussed later in the Discussion section.

4.2.1.2 Training Label Poisoning Attack

The largely negligible performance observed in the training data poisoning attack prompted a thought experiment that the samples were not enough to be poisoned, and that some labels would also need to be poisoned to prevent trends in the images being learnt within a few samples and rounds.

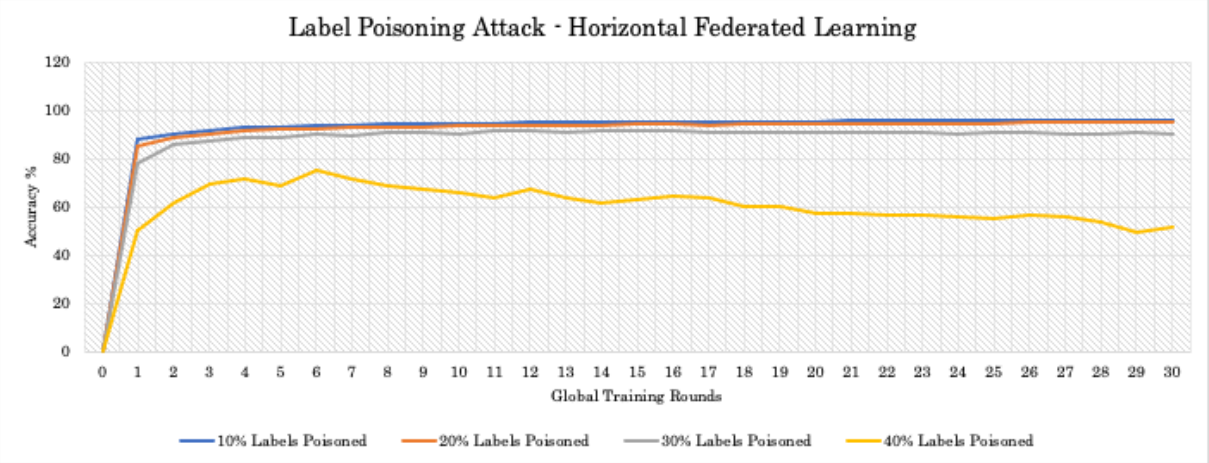

Figure 21

Whilst this label attack is not as uniform as a sample-space attack, it can be seen to have more of an impact on both systems. In the context of Horizontal Federated Learning system (figure 21), 10% and 20% labels poisoned labels have negligible effect as the model still reaches the desired accuracy of 90% within the first round, and then convergence is similar to what was observed in the baseline experiments. At 30% the accuracy is reduced minutely with 90% accuracy being reached in round 3 instead of 1 and then converges to

the expected 95% accuracy mark.

At 40 is where the most effect occurs, with no convergence occurring, and the peak accuracy of 75% at round 6. Then the accuracy begins to deteriorate and settles at the 50% accuracy mark after completing the full 30 global training rounds. This effect is drastic and unexpected, but it is clear to observe that label poisoning has a detrimental effect to the accurate of the model after 40% poisoning, and after 50% it is hypothesised that the model will not be able to give an accuracy higher than a random guess and thus the model becomes unusable.

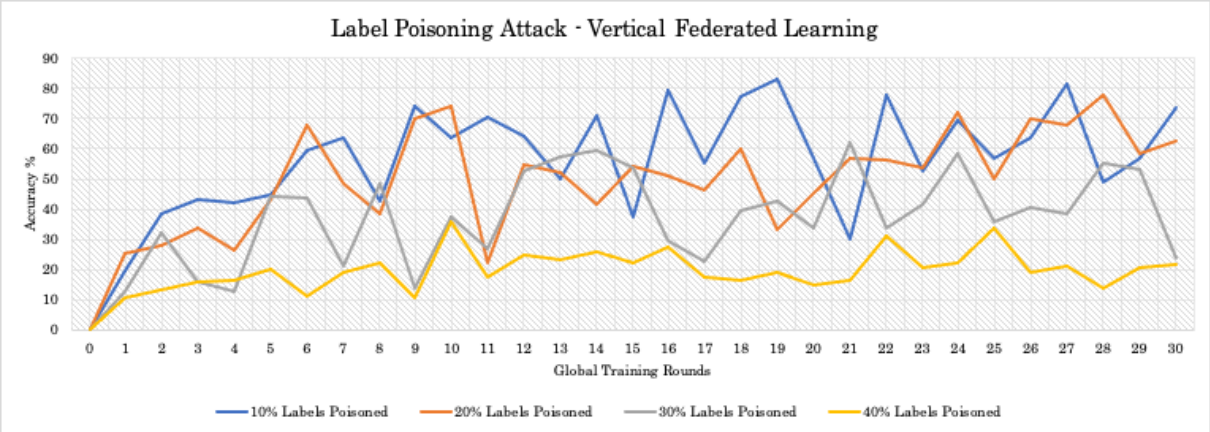

Figure 22

Whilst more sporadic, the Vertical Federated Learning system (figure 22) behaves in a similar manner under the label poisoning attack in terms of accuracy, with the higher the spread of the attack, the lower the average accuracy of the model. However, what is extremely different to the baseline model is that none of the models converge. It is clear that no meaningful result can be extracted from any round of the model for any spread of the attack. For example, in round 20 the accuracy of the 10% spread is over 80%, then 2 rounds later it is nearer 30%. Therefore, the model seems to be effectively outputting guesses with fluctuating accuracy which is an unusable model.

4.2.1.3 Test Data Poisoning The first set of experiments conducted target the test data and poison the test samples. The first experiment conducted was by fixing the potency parameter at 20% and testing different poisoning spreads as per the definition of a "Low Strength Attack". This experiment is designed to see how much of the dataset is required to be poisoned by a low-level potency to reduce the accuracy:
These two attacks step through the strength with a fixed potency of 20%. The experiment starts with 50% of the dataset being poisoned, then escalate to 70% and 90% of the samples being poisoned.

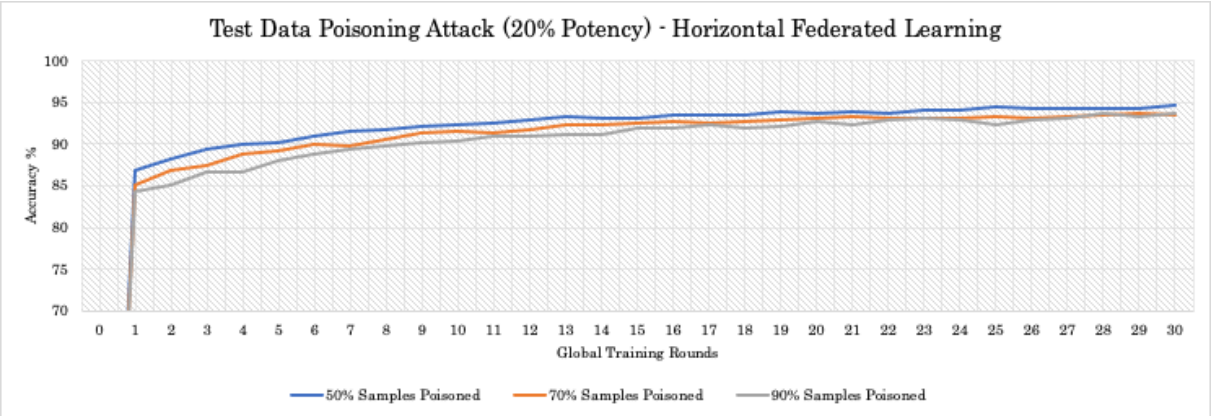

Figure 23

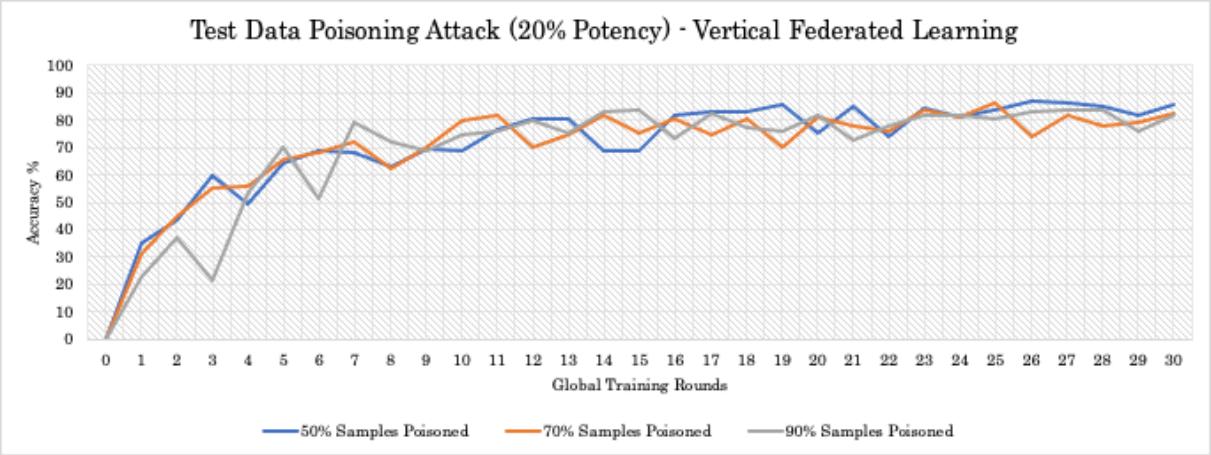

Figure 24

As can be seen by the graphs, there is minimal change under this low potency, and especially with a low number of samples being poisoned. In the case of Horizontal Federated Learning (figure 23), only after 90% of the samples can we see a drop-in accuracy of 3-5% compared to the baseline accuracy, which is not a dramatic effect. The results are similar with Vertical Federated Learning (figure 24) where the accuracy is similar to the baseline, and not much effect can be seen with this 20% potency. Both systems still seem to converge with minor tremors in the accuracy, and reach the expected 90% accuracy within the defined 30 global training rounds.

For the next experiment – the Medium Strength Attack – the attack strength with a higher potency of 40% is used on the same Horizontal and Vertical Federated Learning Systems on this poisoned test data. This experiment is designed to test a higher potencies impact depending on how much of the dataset is affected and the results are as follows: These results are much more dramatic, and feature an immense change of results. In the case of Horizontal Federated Learning (figure 25), it is clear nearly every aspect of the

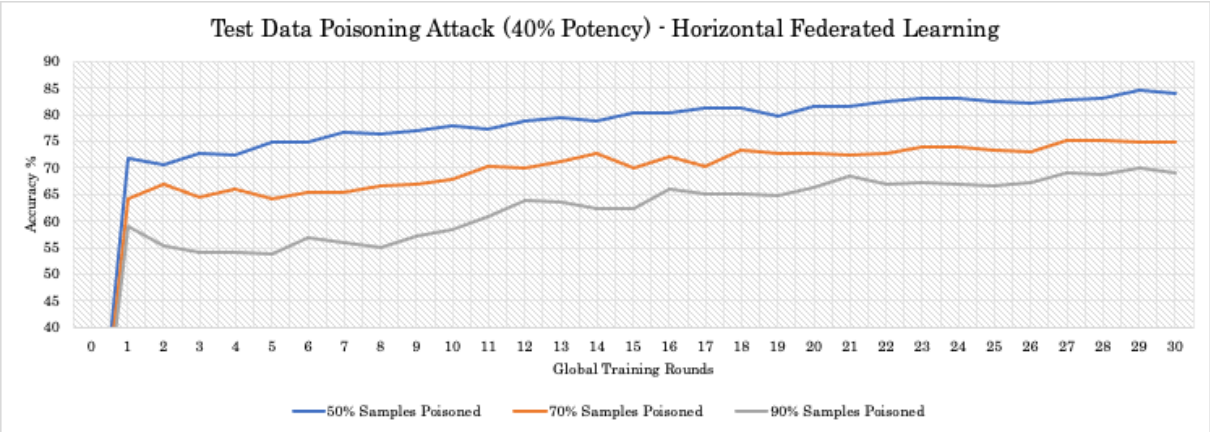

Figure 25

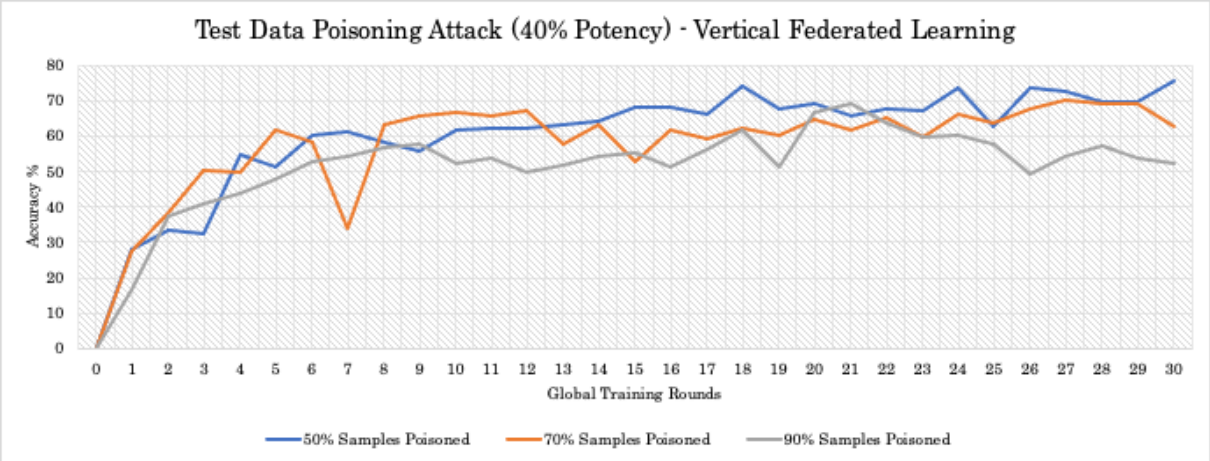

Figure 26

model is affected. Firstly, the high accuracy start of round one that is seen in the baseline no longer occurs, with a start in round 1 in the low 70%'s in the best case and high 50%'s in the worst case instead of the expected 90%. The model then takes a longer number of rounds to converge with little-to-no improvement from round to round, until the final 30th round in which the accuracy is around 85% for the 50% sample attack, or worst case in the mid 60%'s for the 90% sample attack. The Horizontal Federated Learning system never reaches the desired 90% accuracy, and this is considerably lower than the baseline.

The same can be stated for the Vertical Federated Learning system (figure 26), where instead of starting at an accuracy of 30-40%, these models struggle to reach 30% in the best case. The convergence then takes nearly all 30 rounds, with the worst-case scenario of 90% samples actually finishing convergence at 20 rounds, and then seeing a reduction in accuracy over the final 10 rounds. This seems to be far out of character for this system, and it is evident this poisoning attack has had a severe impact even only at 40% potency.

To give a visualisation of the effect of the parameters with real values, a data extraction has been conducted and rendered in the below image (figure 27). The first image (image A) is a benign image that has been poisoned with a potency of 20%, such that all pixels within the image selected have a 20% chance of being poisoned, this is in comparison to the second image (image B) that has been poisoned with a potency of 40% such that all pixels within the image have a 40% chance of being poisoned:

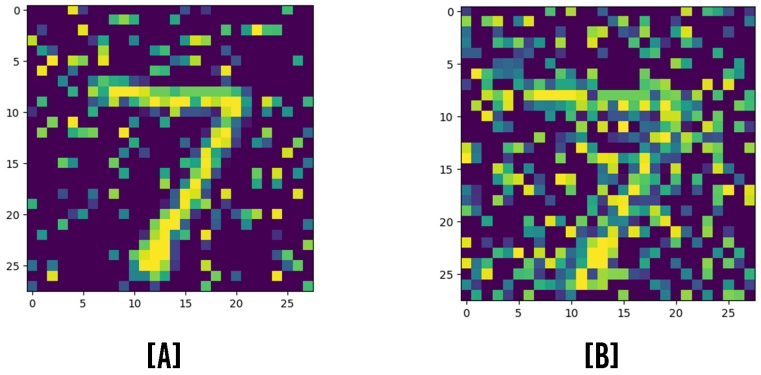

Figure 27

4.2.2 Backdoor Attacks

As discussed previously, a backdoor attack should not lead to any noticeable differences in the model convergence and peak accuracy, and therefore a backdoor experiment is conducted in which 10% and 30% of the dataset is injected with a simple pixel combination of 2 black rows. The idea behind this is to mimic an image border rather than an obvious pixel-pattern. The green in this rendering (figure 28) represents the intensity of black and in the below image the original image of a handwritten '5' is shown in comparison to the poisoned image which has the inject pixel combination in rows 1 and 2:

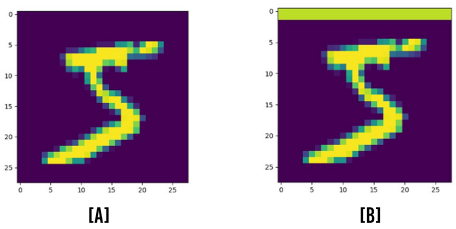

Figure 28

This experiment is effectively a simple pixel-pattern backdoor in which the model will misclassify any image input with the pixel combination described above as a zero regardless of the number in the image.

In order to test that the backdoor does not affect the model in a distinguishable form, an experiment is conducted on both Horizontal and Vertical Federated Learning systems and using a 10% and 30% dataset poisoning rate to check if there was any noticeable anomalies in the accuracy or convergence rate.

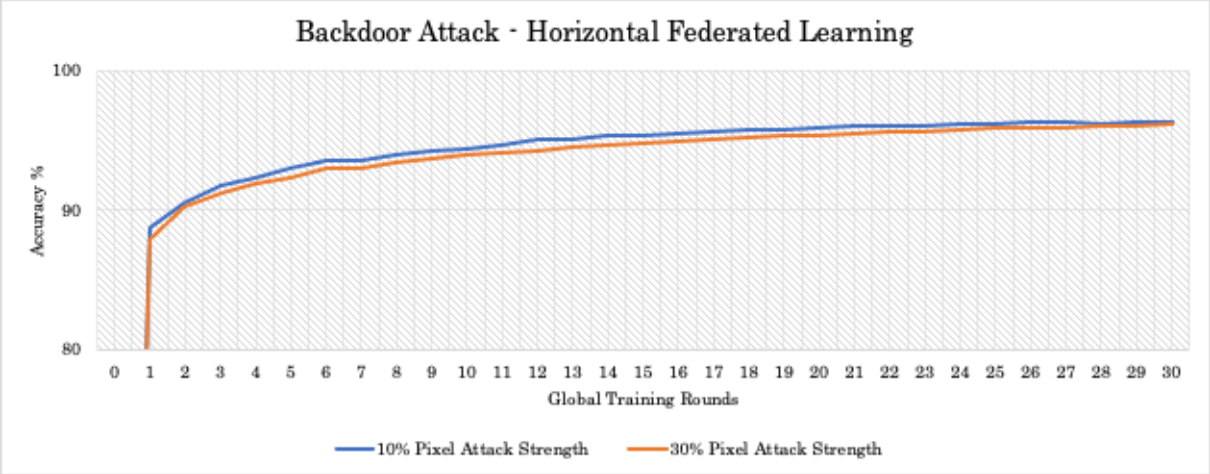

Figure 29

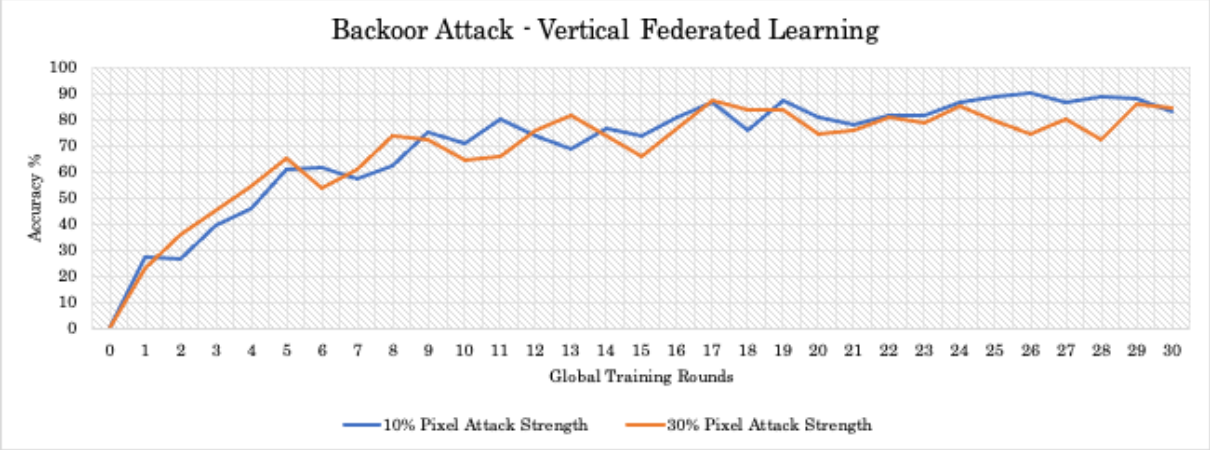

Figure 30

Both Horizontal (figure 29) and Vertical (figure 30) Federated Learning systems converge as expected to the peak accuracy similar to the accuracy represented in the baseline experiment with no noticeable anomalies in the result also as expected. The accuracy of the model remains high for both systems, with a convergence of approximately 85-95%.

The backdoor does not seem to have affected model performance negatively, which was the objective and the increase in pixel attack strength does not seem to follow a different trend. Both systems plot the 10%- and 30%-pixel attack strength as having similar accuracy and therefore it is clear that this attack cannot be detected and model updates will arguably be indistinguishable from genuine local model updates from honest clients.

Note: It is worth noting that since this is an active attack and it takes computational resources to iteratively manipulate the images, this attack is expected to be slower, and whilst time is not usually a measured attribute of models, from running these experiments, it is clear that backdoor attacks take multiple minutes to process the images and inject the pixel combination before training.

In order to test that the backdoor genuinely exists, two validation experiments were conducted on the generated model in which the MNIST dataset was used as an input with 50% of the data samples poisoned with the same pixel-pattern to check if the model misclassifies the backdoor samples:

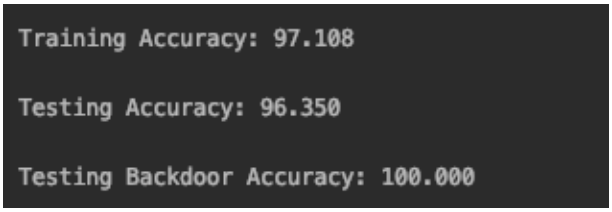

Figure 31

As shown by the experiment (figure 31), the backdoor accuracy was 100% and all the images that were used as inputs were correctly misclassified as 0 and the rest of the images that were unmodified were classified with an accuracy of 96.35% in this horizontal federated learning example. This shows that the benign samples are unaffected, and generally the model operates how it is designed to operate, but in the specific pixel-pattern designed, the model also has the function of classifying those images as a zero.

Another experiment that was devised to check whether the backdoor actually existed was to input a full dataset of images that contained the backdoor pixel-pattern to check the backdoor accuracy. This test is designed to confirm that the backdoor undeniably exists: The experiment (figure 32) shows that the backdoor accuracy was 100% and the model accuracy was 9.8%. This shows that all images with the pixel that are labelled as a zero are correctly accepted by the model as this is what it has also identified, however there seems to be an extra 9.8% of images that are also accepted as benign samples. This is an unintended outcome of the test, and the hypothesis as to why this 9.8% occurs is discussed in the Discussion section.

```
Training Accuracy: 100.000

Testing Accuracy: 9.800

Testing Backdoor Accuracy: 100.000
```

Figure 32

4.2.3 Model Poisoning Attacks

As mentioned in the methodology, the model poisoning attack targets the weights of the model before transmission. There are two distinct adversaries in which a random model update is poisoned (the adversary must own more than one client participating in a pool, and only some are selected to contribute to model updates - also known as a Sybil attack) and the second adversary is one whom only controls the same client that always contributes to model updates. Both of these scenarios are tested as follows:

4.2.3.1 Random Client Selection: Using the standard 10% client participation rate (PR), an experiment is conducted to increment a different percentage of these participants models being poisoned. The impact of 10%, 20%, 30%, 40% and 50% of the participating clients updating the global model with poisoned models is evaluated as follows:

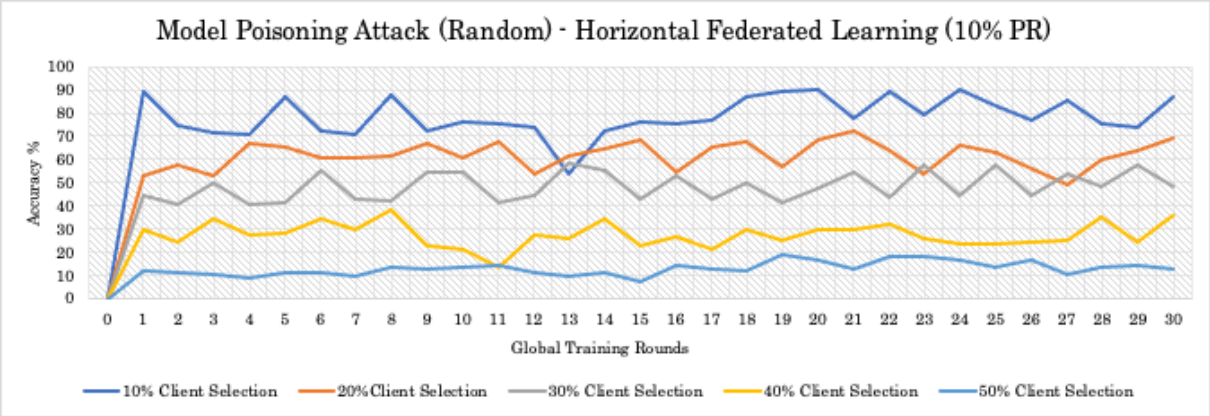

Figure 33

Given the oscillation in the accuracy in figure 33, the average accuracy for each client selection percentage was 78.6%, 61.7%, 48.4%, 27.6% and 12.9% respectively. This evidently shows that the more clients that are selected to be malicious, the lower the accuracy of the aggregated global model. This follows intuition, as the higher the amount of sporadic and random data introduced by the malicious clients, the less honest information the model to create the global model. This is not that much of a drastic effect, as convergence is

still clear and the effect is coherent.

With this negligible result, and given that the parameters are 100 clients with 10% participation, another experiment was conducted to determine if malicious client participation numbers would reduce the accuracy and thus the experiment increased the participation percentage to 30% for this experiment (therefore 30 clients per round transmit updates instead of 10). This experiment was then conducted with the same 10%, 20%, 30%, 40% and 50% of the participating clients are updating the global model with poisoned models on both horizontal and vertical federated learning, and observed the results:

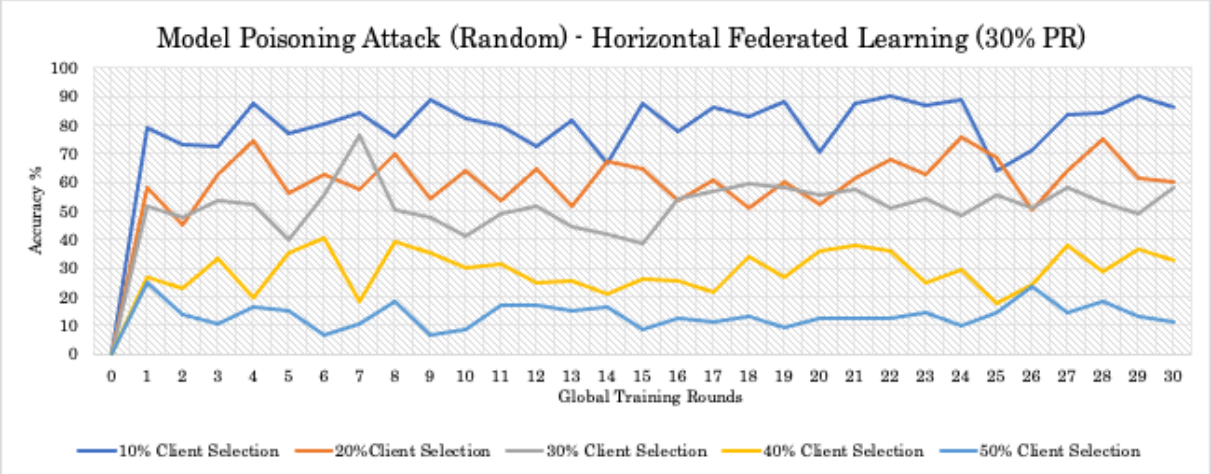

Figure 34

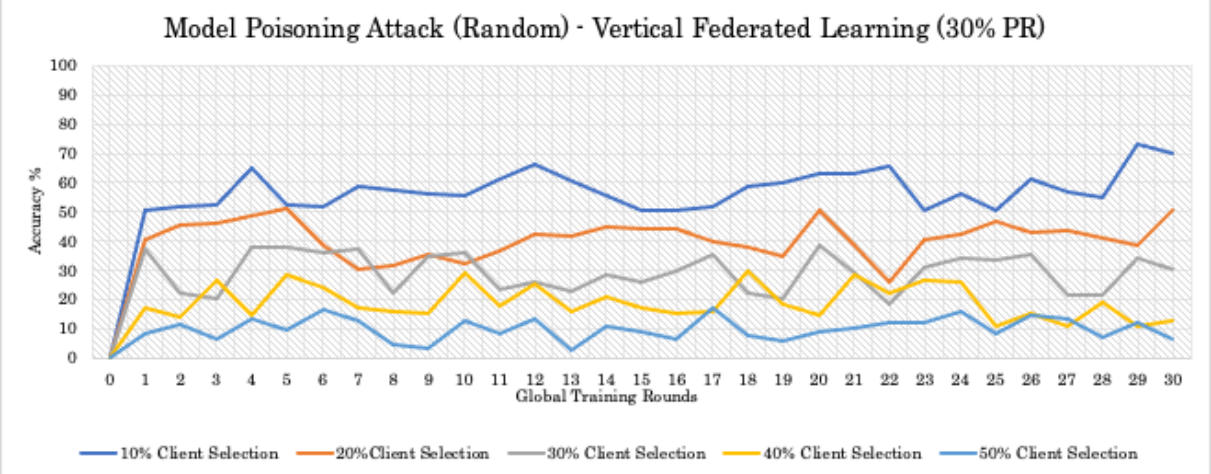

Figure 35

Accuracy is drastically lower than the baseline models now, and the more clients selected, the worst the performance gets. In the Horizontal Federated Learning system (figure 34), the 10% client selection sees a lower accuracy of an average of 80%, however, as the client

selection increases, the accuracy drops, with 50% (or 15/30 participating clients) client selection it is clear that the performance has depleted and this model is no longer useable. This 50% problem is discussed further in the Discussion section as possibly being a well-known problem in distributed systems and networking. The Horizontal Federated Learning system does not converge, and reaches an average accuracy of above 10% at 50% client selection which is so low that a random guess of what the data sample is classified would arguably have the same chance as this depleted model.

The same is reflected in the Vertical Federated Learning system (figure 35), with the more clients selected, the lower the model accuracy, with this system having even worse results at 50% client selection – nearly 5% accuracy and no convergence. It is also odd that the Vertical Federated Learning system seems to be operating at a less sporadic rate than Horizontal Federated Learning, and the low accuracy is instead consistent across all the rounds with little-to-no improvement and no results resembling a close enough curve of convergence.

4.2.3.2 Targeted Model Poisoning: Opposite to Random Client Selection (random Sybil attack), a targeted model poisoning refers to the same set of clients updating the global model with malicious local model updates. An experiment was conducted in which a fixed number of clients are always selected, and the same clients are selected to provide malicious updates.

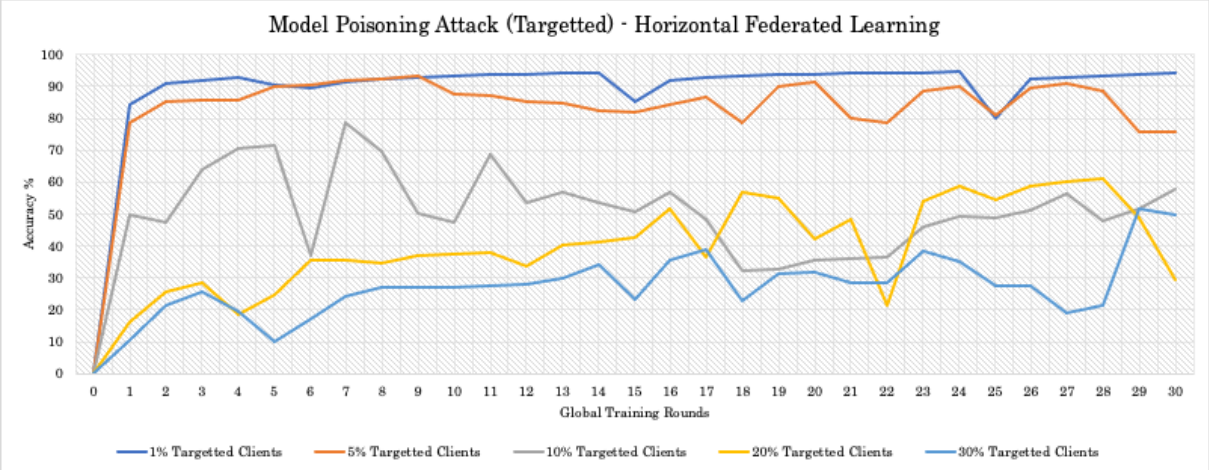

Figure 36

The results of the targeted model poisoning are more difficult to analyse due to the sporadic results and lack of convergence. In the context of Horizontal Federated Learning (figure 36), the 1% malicious client participation seems to have little effect on the accuracy, with 90%+ being reached within 3 rounds, and then converging at nearly 95%. The accuracy takes a drop sporadically, but generally it is a consistent result. This similar for

5% targeted clients but struggles with convergence above 90%. It then drastically drops as the targeted clients increase and more begin to participate in the model updates, the system fails to differentiate between honest and malicious model updates and the correct weightings are diluted with the random updates transmitted.

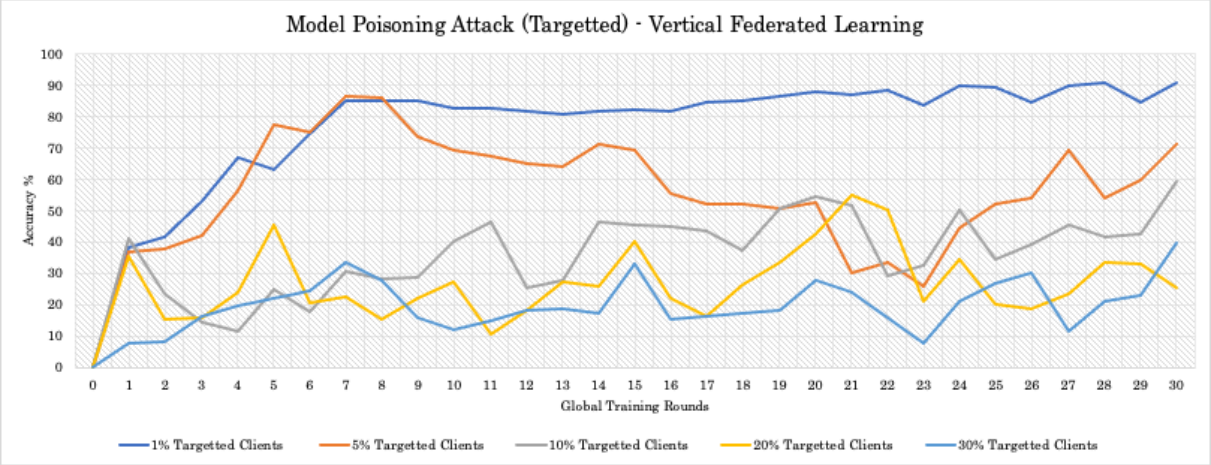

Figure 37

Vertical Federated Learning (figure 37) does not fare well either under this attack. We can evidently observe an attempt of convergence at 1% as the accuracy increases, however, increase peaks at under the 90% target and taking 6 rounds to come close to convergence. At 5% the model rises to a similar peak in the same number of rounds, but as more malicious model updates start to affect the model, the accuracy plummets and cannot be regained. Anything beyond this is sporadic, low-accuracy, non-convergent outputs that are signs of a failed model that should not be used.

Overall, Model Poisoning attacks seems to be extremely strong with a small number of fixed adversarial participants that consistently provide malicious updates with only 5% of the participants needed to see an impact on both Horizontal and Vertical Federated Learning Systems, in comparison with maybe 30 or 40% of randomly selected clients with 30% participation rate to see a strong impact on accuracy.

5 Experiments: Discussion

Beyond the analysis of the results, the following section presents thoughts, hypothesise and comments on the outcomes of the experiments conducted within this paper:

5.1 Performance Experiments

Given the baseline parameters and Federated Learning system design by McMahan et al. [15], it is clear to see that overall Federated Learning works well compared to centralised machine learning. The models converge and the accuracy is high (figure 13) and the models improve as the parameters are optimised with a "good enough" global (figure 14) and local training round numbers (figure 15, and 16). As mentioned in the results analysis, this performance comparison was key to setting a benchmark baseline result that all the experiments can be compared against.

The researchers at Google have evidently created a system that works, and the deployment in the wild on millions of Android devices reinforces this, however from the experiments show, Horizontal Federated Learning clearly outperforms Vertical Federated Learning simply due to the lack of complexity in the sample/label space distribution across clients.

5.2 Data Poisoning Attacks

In general, data poisoning attacks seem to be effective against both Horizontal and Vertical Federated Learning Systems given the spread of the attack is sufficient to poison a large percentage of the dataset, and the potency of the attack is sufficient enough to make the image unreadable such as 40% potency as shown in (figure 19 and 20).

In the ***training data poisoning*** (4.2.1.1), both Horizontal and Vertical Federated Learning system react negatively to the introduction of poisoned data, but based on the test 40% potency is still not effective, and proven by the off-centre 99% poisoned sample attack, it is proven that the potency of the attack seems to be the main decider on the success of the adversary. The data poisoning attack seems to be effective depending on the potency and how much of the dataset is affected which is equal to the attack strength formula this dissertation presented earlier. It is clear to infer that the higher the number of data samples poisoned and the more poisoning within the samples, the harder it is for the system to create a relationship with the data and thus a lower accuracy.

It is worth noting that the potency is also where most of the computational processing is conducted as the adversary may manipulate a percentage of 784 pixel per image selected, and thus it is clear that the higher the potency the stronger the attack, and the lower the accuracy, but this also requires significant processing capabilities of the adversary –

possibly even setting up a cloud-hosted solution to offload images to perform the data poisoning attack before contributing to the Federated Learning system if the adversary was truly on a mobile device with low computational resources.

Additionally, while not tested due to the fact that testing every value of every parameter in every attack is extremely time consuming, the expectation would be that any potency above 50% or 60% would yield to a drastic effect and possible denial-of-service using this data poisoning attack on the training data. This could have been defined in the methodology as a "High Strength Attack" but it was not defined nor implemented as the computational resources required to test this were beyond the resources available, and the time required to poison 50% or 60% of the 784 pixel multiplied by the 60,000 images in the dataset would have been too much.

An special event that was flagged during testing was the lacklustre performance of the off-centre 99% poisoned sample attack at the 40% potency. The model still seemed to converge correctly and only with a reduction of 5% accuracy even though 99% of the samples featured poisoned data. The working hypothesis for this is that the MNIST data consists of images with relatively basic patterns such as lines and small curves to represent numbers. For this reason, I believe that the model can be trained to learn the classifications with a relatively low amount of input data to a high percentage of accuracy, whereas the model may learn little-to-no information from the random looking poisoned images, and therefore the 1% benign images become the actual sample set as they are the only images that the model can pick up trends from.

It should be noted that the MNIST train dataset contains 60,000 training pairs. Therefore, even if, an extremely large percentage of training pairs are poisoned, say 60%, there will still be 26,000 images that have not been poisoned, and I would argue that the model can still be trained to high accuracy with these 26,000 images and even arguably less. The 34,000 images that are poisoned will not have any real patterns for the model to pick up on during training and therefor seem to contribute a negligible amount of "learning" the model actually conducts, and thus these images will not negatively affect the training of the model.

This theory is backed up by the concept of Federated Learning too. As the 60,000 images are split horizontally to 100 clients, each client has 600 images. Each client then locally builds a model successfully, with only these images which means that only 1% is actually needed for the global model in aggregate. Thus, the working theory is that even if 99% of the samples are attacked, there should be some decent performance from the model.

Label poisoning (4.2.1.2) is where results become arguably more interesting. Since the

dataset is made up of samples (784 pixels) and an associated label, it seems feasible that if any adversary had the ability to change a sample, they could also change the label, and therefore it was worth exploiting as an attack vector. Most papers within the Federated Learning do not explore the effects of label poisoning, given that the results presented show that label poisoning is effective on 40% of the labels in the Horizontal setting, and even less in the Vertical setting at 20-30%. Label poisoning is also advantageous for an adversary as data protection mechanisms are arguably stronger when more data is concerned, and therefore the less data an adversary has to change (i.e. 40% of single digit labels vs 40% of 600 images of 784 pixels) the computationally easier to conduct, with less chance of being caught.

An open question is the sudden effect of the 40% label attack, in comparison to 10%, 20% and 30%. I am unsure as to why this attack suddenly came to life at 40% of the samples targeted, however, a working theory is similar to the 99% attack mentioned above. The model is able to learn a lot of from the benign samples and override the patterns learned from the poisoned labels, and therefore once it gets to a certain percentage – in this case 40% - the effect of the poisoned label no longer becomes negligible, and the model arguably becomes confused. For example, it is learning that an image with the number 6 is labelled as a 6, but when 40% of the data says that an image with a number 6 is a 4, it is inevitable that the accuracy will be reduced.

Changing labels as an attack is debatable, yet feasible. The concept seems to work depending on how many labels are attacked, but I would argue that there is a fine line. If the labels are changed for a percentage of samples, it makes sense that the model accuracy would reduce. For example, if the model is picking up that an image of a 6 is labelled as a 6, then a similar image of a 6 is labelled as a 4, the model weighting are sure to be affected. The fine line arguably comes into effect when the majority of labels are changed, for example, the model is given an image of a 6 and then labelled as a 7, then another image of a 6 is also labelled as a 7, the model will actually gain accuracy. This will obviously be accuracy towards the wrong number, but the accuracy will still increase, as the mislabelled trends in the data become too coherent.

Test data poisoning (4.2.1.3) is definitely one of, if not the weakest attacks presented, even with a Medium Strength attack targeting 90% of the samples with a 40% attack potency (figure 25 and 26), the results are clear to see that whilst the results do appear dramatic in this instance, placing this result contextually and comparably with the other test results (e.g. model poisoning), the attack does not provide much change to the accuracy. Additionally, the amount of computational resources required to process 600 images per client and manipulate a percentage of the 784 pixels is considerable, and thus arguably the effort-reward ratio is one of the lowest. If the adversarial goal is to reduce

the accuracy, this is not the optimal route to take, and complete denial-of-service would arguably need 70%+ potency only possibly 100% of the the samples, which extremely unfeasible.

Overall, its clear data poisoning attacks are a force to be reckoned with. Data poisoning attacks comes in many forms, and this paper had demonstrated that the attack is not as straight-forward for an adversary may initially seem. Training data is arguably the most effective form of data poisoning attack due to the ability of an adversary to tailor the attack strength depending on their computational resources, using the attack strength formula presented earlier in this paper. Label poisoning seems to be a valid and stronger attack vector, but the logic on why the labels can be be poisoned and what effect that has on the overall model is still debatable given the attack strength and which labels are attacked, thus more experiments could be conducted on this. And finally, the weakest form of data poisoning is the test data poisoning. This bare has any effect on the model, and any effect it does have comes from a "validation" perspective such that the model is not even affected, only the process of testing the model is.

Therefore, I would conclude the discussion on data poisoning attacks as conclusive if the poisoning attack mounted by an adversary is a training data poisoning attack or possibly a label poisoning attack, and therefore these attacks should definitely be considered as credible threats when implementing federated learning in the wild - as shown by the presented results.

5.3 Backdoor Attacks

The ***backdoor attack*** (4.2.2) is arguably the strongest form of attack since the objective is to introduce a function to the model that only the adversary can access. This form of attack is covert and extremely effective.

The attack clearly succeeded in introducing the backdoor for both Horizontal (figure 29) and Vertical (figure 3) Federated Learning, and this is what is interesting about the comparison of the test results with the baseline performance results initially conducted. The backdoor is indistinguishable looking at the accuracy, the time taken or the loss on each round, yet the backdoor exists. An adversary could be using the backdoor to get an image-processing system to accept forged documentation, and the metrics do not reveal anything about this occurring.

In the test case in which all test samples were images with the pixel-pattern injected (figure 32), it was expected that the testing accuracy of the model would be near 0%, however in reality it was 9.8%. This was an odd observation contrary to the expected results. However, upon further examination the working hypothesis to this is that all images

injected with a pixel-pattern are classified as a Zero, and this images with a handwritten zero are injected with the pixel-pattern just as any other image, and then misclassified as zero as this is the backdoor misclassification task. However, "misclassifying" a zero as a zero still outputs the correct classification and therefore the only samples in the dataset that are tested correctly by the model are zero's as this is the coincidental number selected for the backdoor. If the misclassification output was set to a character that is not [0-9] then no images would be classified correctly and I expect the model accuracy to be 0% whilst the backdoor accuracy be 100% in this case.

Whilst backdoors are dynamic and very advantageous for an adversary, as defined in the methodology, the spread of the attack needs to be just right otherwise it does not work. This is a condition only applicable to backdoor attacks, and therefore backdoors require some intelligence on the part of the attacker to define a subtle-yet-distinguishable pixel-pattern, and select the correct amount of data samples too attack such that there are enough poisoned samples for the model to pick-up the features of the pixel-pattern, but not too much such that benign samples are not enough and the model is only learning from poisoned samples.

Additionally, there is an attack type named "single-shot attack" by Bagdasaryan et al. [1] which this dissertation did not anticipate to evaluate, however from the results of the tests presented previously in this paper (figure 29), it is clear that a single-shot attack is possible as the accuracy of the main model is not reduced at any of the rounds, including the first training round. Therefore, as a side note, this paper accepts the notion of single-shot attack and an independent test case with methodology should be created. However, as a proof of concept, the Backdoor Horizontal Federated Learning experiment was conducted, but with the global training round set to 1, and the global model nearly reached an accuracy 90% which is expected, but what is interesting, is that the backdoor attack accuracy was 100% in the first round (figure 38). Therefore, based on this preliminary experiment, it is highly likely that single-shot attacks are also a feasible attack vector.

Overall, if the adversarial goal is to avoid detection, and also maliciously benefit from the deployed model, backdoor attacks seem like the perfect fit. As demonstrated and discussed, with a relatively low amount of input from the adversary, the models achieve a near perfect accuracy on the backdoor task, in addition to the fact that anomaly detection and observing the behaviour of the model output is sure to lead to the observation that the model is behaving correctly, when in fact it is being used to conduct malicious activity.

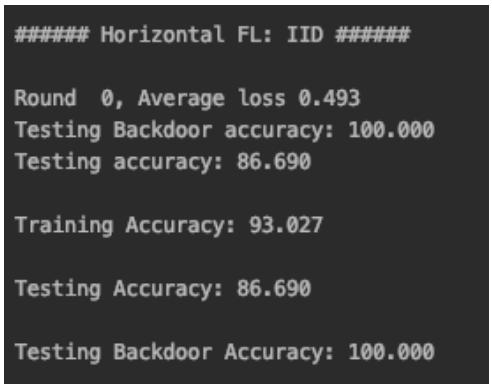

Figure 38

5.4 Model poisoning

Model poisoning is arguably the most studied attack vector as it is also the attack that requires the least work on the behalf of the adversary as only the weights need to be sent incorrectly to the aggregating server. Additionally, as indicated by the experimental results it is also clearly the most powerful attack if the adversarial goal is model accuracy reduction or total denial-of-service. This is consistent with the views of other researchers such as Lyu et al.[14] and Bhagoji et al.[2] whom also argues that model poisoning is far more effective than data poisoning.

In the case of ***random client selection*** which is a form of Sybil Attack, and based on the results presented, it is clear to see that the higher number of malicious clients that the adversary has control over, the lower the accuracy. On average, for every 10% of clients the adversary controls, the accuracy is reduced by 20%, which is a staggering effect especially if the client number is small such as 100 clients.

It is worth noting however that the number of malicious clients an adversary control is not always representatives of how many clients in a particular training round are selected as this is random. I believe this is the cause of the fluctuation in accuracy seen in the graphs presented for this attack (figure 34 and 35), as some rounds may feature more/less malicious clients and more/less thus malicious updates.

One additional comment that is clear to see also is the effect of the 50% client selection. An overpowered adversary with access to 50% of the clients in the system reduces the accuracy to under 10% and this is unacceptable for a model to operate. This is interesting and if some abstract thinking is applied, it would be clear to see that there are similar problems in other aspects of distributed systems such as blockchain systems in which a similar problem defined as “the 51% problem”. This is where an adversary control 51% of the participating nodes in the block-chain system, and can then fake inputs to the system

and verify them by themselves as they are the most powerful party in the system.

Whilst this scenario is not completely and directly applicable here, I would argue that the 51% problem evidently exist as I believe any more malicious model updates above 51% will effectively lead to a complete denial-of-service, which is one of the possible adversarial goals.

Targeted Model Poisoning attacks operate slightly differently, and this is arguably represented in the results as well (figure 36 and 37). This form of attack relies on a repetitive malicious model updates from the same set of clients that are controlled by the adversary, and thus it is guaranteed that however many clients the adversary owns, they will participate in the update round. The effect of this can be seen when the test is capped at 30% client selection and the model operates at 10%-20% accuracy, compared to the required 50% of random client selection to achieve the same disruption to the model.

Overall, model poisoning seems to be effective against both Horizontal and Vertical Federated Learning and dependant on the number of clients, whether a single attach or a Sybil attack. This is consistent with results seen in other papers [14] [6] [3], and therefore the results clearly show that model poisoning is a valid attack vector and must be considered carefully when implemented Federated Learning in the wild.

6 Conclusion

Overall, the test experiments designed and presented in the previous sections successfully survey and test the attack research published by the aforementioned researchers in the Federated Learning research sphere. Whilst the discussion section presented thoughts on the results of the results, we must evaluate the success of the paper as a whole, and contribution of the paper to the rapidly evolving field of Federated Learning:

6.1 Goal Evaluation

Most papers tend to evaluate the success of a project based on the contribution, but I believe that a more suitable metric of evaluation on the success of a paper that provides a survey and implementation is to compare the outcome of the paper to the initial goals that are defined in the introduction:

- The ***first goal*** was to effectively conduct a survey into the current literature, but not only present was relevant to this project. The survey conducted collates the top papers in the Federated Learning research scene, and collates overlapping content into a single "unified" literature review. This is effective for researchers as it arguably acts as a check point in the research timeline, and helps research develop faster but also with more depth. Thus, this goal has been achieved successfully.

- ***Goal two*** is more difficult to evaluate on success. A baseline federated system was built as a sandbox in which the attacks could be mounted, and this is the reason the presented results and discussion section is so effective, is that different attacks and adversarial goals become directly comparable when the system they are attacking is the same. This is one of the distinguishing factors of this papers, and arguably the reason ***Goal Three*** exists as a goal. This was completed and implemented throughout the attacks as attack parameters, which have not been discussed in the papers aforementioned in the literature review.

- The systematic analysis and comparison mentioned in ***Goal Four*** is arguably the underlying premise of this paper. This was designed in the experiments conducted, and hence why multiple parameters and experiments were conducted for both Horizontal and Vertical Federated Learning. The results of these experiments are arguably one of the main contributions of this paper, and provide a lower-level insight into the majority of security attacks against Federated Learning for would-be implements to be aware of, or parties interested in the security of the systems deployed on their devices.

Overall this paper contributes many different facets of research, from a unified survey of the current state-of-art in Federated Learning, to defining/implementing attack parame-

ters and unified nomenclature relevant to most security attacks presented, to a discussion of these results and their true impact on Federated Learning from an adversarial perspective.

6.2 Limitations

Whilst all papers have flaws, it is a learning opportunity to identify the limitations of the paper in the hopes that this can further the research by inspiring further work on the limitations of this paper:

- Most security experiments conducted are limited in scope, and only go up to a certain attack strength. For example, the data poisoning attacks are 50%, 70% and 90%. This was designed to save time on limited computational resources, and therefore if there was cloud access for further work, this limitation could be removed and all the parameters tested.
- This paper selected a scope and focused only on Security attacks, which is half of the possible attack vectors. A complete evaluation would have evaluated and implemented privacy attacks aforementioned in the background section, however this would have been arguably beyond the scope of this dissertation.
- Given the current state of the world and the current pandemic, the computational resource access is limited, and therefore all these experiments were conducted on a personal machine from 2014. This limited the number of tests conducted, and certain parameters such as the number of users in the system.
- The backdoor attack only defines one set of pixel-patterns due to the computational resources required to run image processing algorithms as simple pattern was selected. A more discrete but complex pattern could be run on the cloud in a fraction of the time and thus different patterns could be tested.

6.3 Future Work

In order to progress the unification research in Federated Learning, the following research must be conducted:

- Increased depth into the parameters of the attacks with higher computational resources, but more importantly on real devices in the wild.
- This paper only ventured into security attacks, and therefore I would recommend that a unification and testing paper is created similar to this one for the privacy attacks, in which different attacks and parameters are tested for effectiveness for both Horizontal and Vertical Federated Learning.

- A further paper could be created on the unification of defence theory by applying what has already been researched and implementing it just as this paper has done. This will create a clear "go-to" set of papers for attacks and defences rather than 10's of random papers on sub-topics.

- These unification papers, similar to this one, are effective in grouping disparate research releases, and therefore another version of this paper could be created the future (e.g. 1 year from now, or 3 years from now) that evaluates this paper, and then re-evaluates the same attacks if they are still relevant, and new attacks that are developed since the release of this paper.

- To address the limitations mentioned of the Backdoor experiments, a future paper could be designed such that it addresses how effective the attack is on different parameter, similar to the structure of this paper. There are multiple papers, including this one, that prove backdoor attacks are feasible, but to extend this paper it would be useful to know answers to questions such as how much of the image needs to be poisoned (e.g. 10 pixels, 5 pixels, 1 pixel?), positioning of this pixel (e.g. middle, bottom left, etc.) and how many images are needed for this to work.

6.4 Concluding Remarks

This paper set out to achieve certain goals which have presented a comprehensive literature review on the current state of federated learning research within the scope of security and privacy attacks. Additionally, an implementation of Horizontal and Vertical federated learning has been adapted and attacked by custom attacks, in which this paper has presented new and different parameters to the attacks. And finally, an analysis of the effectiveness of the attacks against both Horizontal and Vertical Federated Learning has been conducted, in which a discussion has also been conducted which hypothesises the reason for the results and possible adversarial advantages/disadvantages that were not clear before the implementation had been conducted.

Overall, this paper has much to contribute to the field of Machine Learning in general, and more specifically the security research field of Federated Learning in which this paper provides great insight into the true effects of the attacks commonly mentioned, and thus the goals have been met. This paper also presents new parameters to define attack strength, and also justifies the need for more unification papers that implement attacks on the same systems so that results can be directly compared. This paper also proves that all the attacks mentioned are feasible for different levels of adversaries, with varying objectives and computational resources, and thus when Federated Learning is implemented these attacks must be taken seriously to prevent denial-of-service or unintended use-cases of the developed machine learning models.

7 Appendix

7.1 Implementation Code

Backdoor Attack Snippet:

```
def BackdoorTrain(dataset_train):
    args = args_parser()
    for i in range(len(dataset_train)):
        if args.Backdoor1 >= random.choice(range(1, 101)):
            for j in range(2):
                for x in range(28):
                    dataset_train[i][0][0][j][x] = 2.8
            dataset_train[i][1] = 0
    return dataset_train

def BackdoorTest(dataset_test):
    for i in range(len(dataset_test)):
        for j in range(2):
            for x in range(28):
                dataset_test[i][0][0][j][x] = 2.8
        dataset_test[i][1] = 0
    return dataset_test
```

Training Data Poisoning Attack Snippet:

```
def dataTraimPoison(dataset_train):
    args = args_parser()
    percPair = args.DPAtrain1
    percJustLabelorInput = args.DPAtrain2 # label and input or just one
    percLabelorInput2 = args.DPAtrain3 # label or input
    percPixel = args.DPAtrain4 # chance of pixel being changed when looping
        through input data

    final = 0
    finalL = 0
    finalP = 0
    attackedP = []
    attackedL = []
    for i in range(len(dataset_train)):
        if percPair >= random.choice(range(1, 101)):
```

```
            final = final + 1
            # chance of just label or input changing or both changing changing
            if percJustLabelorInput >= random.choice(range(1, 101)):
                # only one or the other changes
                if percLabelorInput2 >= random.choice(range(1, 101)):
                    # label changes
                    old = dataset_train[i][1]
                    while old == dataset_train[i][1]:
                        dataset_train[i][1] = random.choice(range(0, 9))
                    finalL = finalL + 1
                else:
                    # input changes
                    finalP = finalP + 1
                    attackedP.append(i)
                    for j in range(len(dataset_train[0][0][0])):
                        for x in range(len(dataset_train[0][0][0][j])):
                            if percPixel >= random.choice(range(1, 101)):
                                dataset_train[i][0][0][j][x] = random.random() *
                                    2.8215
            else:
                # both change
                # label
                old = dataset_train[i][1]
                while old == dataset_train[i][1]:
                    dataset_train[i][1] = random.choice(range(0, 9))
                # input
                attackedP.append(i)
                for j in range(len(dataset_train[0][0][0])):
                    for x in range(len(dataset_train[0][0][0][j])):
                        if percPixel >= random.choice(range(1, 101)):
                            dataset_train[i][0][0][j][x] = random.random() *
                                2.8215


    plt.imshow(dataset_train[attackedP[0]][0][0])
    plt.savefig('./Graphs/attackedImage1.png')
    imageio.imwrite('./Graphs/attackedImage.png',dataset_train[attackedP[0]][0][0])
    return dataset_train
```

Test Data Poisoning Attack Snippet:

```
def dataTestPoison(dataset_test):
    args = args_parser()
    percPair = args.DPAtest1
    percJustLabelorInput = args.DPAtest2 # label and input or just one
    percLabelorInput2 = args.DPAtest3 # label or input
    percPixel = args.DPAtest4 # chance of pixel being changed when looping
        through input data

    final = 0
    finalL = 0
    finalP = 0
    attackedP = []
    attackedL = []
    for i in range(len(dataset_test)):
        if percPair >= random.choice(range(1, 101)):
            final = final + 1
            # chance of just label or input changing or both changing changing
            if percJustLabelorInput >= random.choice(range(1, 101)):
                # only one or the other changes
                if percLabelorInput2 >= random.choice(range(1, 101)):
                    # label changes
                    old = dataset_test[i][1]
                    while old == dataset_test[i][1]:
                        dataset_test[i][1] = random.choice(range(0, 9))
                    finalL = finalL + 1
                else:
                    # input changes
                    finalP = finalP + 1
                    attackedP.append(i)
                    for j in range(len(dataset_test[0][0][0])):
                        for x in range(len(dataset_test[0][0][0][j])):
                            if percPixel >= random.choice(range(1, 101)):
                                dataset_test[i][0][0][j][x] = random.random() *
                                    2.8215
            else:
                # both change
                # label
                old = dataset_test[i][1]
                while old == dataset_test[i][1]:
                    dataset_test[i][1] = random.choice(range(0, 9))
                # input
                attackedP.append(i)
```

```
                for j in range(len(dataset_test[0][0][0])):
                    for x in range(len(dataset_test[0][0][0][j])):
                        if percPixel >= random.choice(range(1, 101)):
                            dataset_test[i][0][0][j][x] = random.random() *
                                2.8215

    plt.imshow(dataset_test[attackedP[0]][0][0])
    plt.savefig('./Graphs/attackedTestImage1.png')
    imageio.imwrite('./Graphs/attackedTestImage.png',dataset_test[attackedP[0]][0][0])

    return dataset_test
```

Federated Averaging Snippet: [15][8]

```
def FedAvg(w):
    w_avg = copy.deepcopy(w[0])
    for k in w_avg.keys():
        for i in range(1, len(w)):
            w_avg[k] += w[i][k]
        w_avg[k] = torch.div(w_avg[k], len(w))
    return w_avg
```

Targetted Model Poisoning Attack Snippet:

```
targets = []
    if args.MPAtarget:
        num_targetted = math.floor((args.MPAt1 / 100) * args.num_users)
        for i in range(num_targetted):
            flag = 1
            while flag==1:
                targ = random.choice(range(0, args.num_users))
                if targ not in targets:
                    targets.append(targ)
                    flag = 0
```

Random Model Poisoning Attack Snippet:

```
def ModelAttack(w):
    for i in w.keys():
```

```
3         w[i] = torch.randn(w[i].size())
4     return w
```

References

[1] Eugene Bagdasaryan et al. *How To Backdoor Federated Learning*. 2019. arXiv: 1807.00459 [cs.CR].

[2] Arjun Nitin Bhagoji et al. *Analyzing Federated Learning through an Adversarial Lens*. 2019. arXiv: 1811.12470 [cs.LG].

[3] D. Cao et al. "Understanding Distributed Poisoning Attack in Federated Learning". In: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. 2019, pp. 233–239. DOI: 10.1109/ICPADS47876.2019.00042.

[4] Shaoqi Chen et al. "FL-QSAR: a federated learning based QSAR prototype for collaborative drug discovery". In: *bioRxiv* (2020). DOI: 10.1101/2020.02.27.950592. eprint: https://www.biorxiv.org/content/early/2020/02/28/2020.02.27.950592.full.pdf. URL: https://www.biorxiv.org/content/early/2020/02/28/2020.02.27.950592.

[5] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[6] David Enthoven and Zaid Al-Ars. *An Overview of Federated Deep Learning Privacy Attacks and Defensive Strategies*. 2020. arXiv: 2004.04676 [cs.CR].

[7] Stephen Hardy et al. *Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption*. 2017. arXiv: 1711.10677 [cs.LG].

[8] Shaoxiong Ji. *Federated Learning*. https://github.com/shaoxiongji/federated-learning. 2020.

[9] Shaoxiong Ji et al. "Learning Private Neural Language Modeling with Attentive Aggregation". In: *2019 International Joint Conference on Neural Networks (IJCNN)* (July 2019). DOI: 10.1109/ijcnn.2019.8852464. URL: http://dx.doi.org/10.1109/IJCNN.2019.8852464.

[10] Jakub Konečný et al. *Federated Learning: Strategies for Improving Communication Efficiency*. 2017. arXiv: 1610.05492 [cs.LG].

[11] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (May 2020), pp. 50–60. ISSN: 1558-0792. DOI: 10.1109/msp.2020.2975749. URL: http://dx.doi.org/10.1109/MSP.2020.2975749.

[12] X. Liao, L. Ding, and Y. Wang. "Secure Machine Learning, a Brief Overview". In: *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement - Companion*. 2011, pp. 26–29. DOI: 10.1109/SSIRI-C.2011.15.

[13] Yang Liu et al. “A Secure Federated Transfer Learning Framework”. In: *IEEE Intelligent Systems* 35.4 (July 2020), pp. 70–82. ISSN: 1941-1294. DOI: 10.1109/mis.2020.2988525. URL: http://dx.doi.org/10.1109/MIS.2020.2988525.

[14] Lingjuan Lyu, Han Yu, and Qiang Yang. *Threats to Federated Learning: A Survey*. 2020. arXiv: 2003.02133 [cs.CR].

[15] Brendan McMahan et al. “Communication-Efficient Learning of Deep Networks from Decentralized Data”. In: ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 1273–1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[16] N. Papernot et al. “SoK: Security and Privacy in Machine Learning”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. 2018, pp. 399–414. DOI: 10.1109/EuroSP.2018.00035.

[17] Reza Shokri et al. *Membership Inference Attacks against Machine Learning Models*. 2017. arXiv: 1610.05820 [cs.CR].

[18] Ziteng Sun et al. *Can You Really Backdoor Federated Learning?* 2019. arXiv: 1911.07963 [cs.LG].

[19] Hongyi Wang et al. “ATOMO: Communication-efficient Learning via Atomic Sparsification”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018, pp. 9850–9861. URL: https://proceedings.neurips.cc/paper/2018/file/33b3214d792caf311e1f00fd22b392c5-Paper.pdf.

[20] Qiang Yang et al. *Federated Machine Learning: Concept and Applications*. 2019. arXiv: 1902.04885 [cs.AI].